Time and Resource Constrained Scheduling, with Applications to Space Station Planning

by

Clifford Roger Kurtzman

B.S. University of California at Los Angeles (June, 1981)
S.M. Massachusetts Institute of Technology (February, 1984)

Submitted to the Department of
Aeronautics and Astronautics
in Partial Fulfillment of the
Requirements for the
Degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February, 1988

Signature of Author ______________________________
Department of Aeronautics and Astronautics

Certified by ______________________________
Prof. David L. Akin
Committee Chairman
Department of Aeronautics and Astronautics

Certified by ______________________________
Prof. Rene Miller
Department of Aeronautics and Astronautics

Certified by ______________________________
Prof. Robert Simpson
Department of Aeronautics and Astronautics

Accepted by ______________________________
Prof. Harold Y. Wachman
Chairman, Departmental Graduate Committee
Department of Aeronautics and Astronautics

TIME AND RESOURCE CONSTRAINED SCHEDULING, WITH APPLICATIONS TO SPACE STATION PLANNING

by

CLIFFORD ROGER KURTZMAN

Submitted to the Department of Aeronautics and Astronautics
on January 8, 1988 in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Aeronautical and Astronautical Engineering

ABSTRACT

Past spaceflight experience has shown that current methodologies for performing crew activity scheduling will not be sufficient to support future space station requirements. Current practices for forming Space Shuttle schedules are highly man-hour intensive, and are not amenable to rapid dynamic replanning in the presence of unanticipated or unforeseen events. There is also no way to include the preferences of the crewmembers (for example, the tasks they wish to perform, and when they wish to perform them) into the scheduling process. Further, presently used methods do not incorporate the capabilities of mathematical operations research techniques to perform optimization of schedules in order to maximize the completion of mission goals and find tradeoffs among factors such as differing crewmember skill levels. Scheduling problems have frequently plagued the Space Shuttle, which only operates for periods of a week at a time; on a continuously manned space station in the 1990's, current practices will be totally unacceptable.

In order to bridge these difficulties, this thesis develops an iterative scheduling technique which attempts to search (span) the combinatoric solution space in an intelligent manner. The technique examines (and makes perturbations upon) successive schedules in an attempt to find progressively better solutions; it thus avoids becoming trapped into some fixed scheduling. The technique is also flexible enough to accomodate highly complicated constraint environments; the search inherently focuses upon the parts of a schedule which make it complicated, while avoiding dealing with factors which are not present in a particular problem instance. Typical results using the iterative algorithm reduced solutions to within 4% of optimum in scenarios where previous methodologies (when applicable) would produce solutions approximately 20% worse than optimum.

A methodology is developed to simplify the scheduling problem by using inferences to propagate interrelated time constraints, and thus simplify the scheduling problem. A new heuristic technique, the maximum compatibility method, is also developed to aid in the scheduling of problems which are dominated by the competition for scarce resources.

In order to demonstrate the propagation of time constraints and the iterative algorithm, an interactive computer scheduling tool, known as the MFIVE Crew Activity Planner, was developed. MFIVE provides a user friendly interface for building, solving, and displaying scheduling problems, as well as for investigating the features which will be necessary to eventually provide a real-time scheduler for use on a space station.

Thesis Supervisor: Dr. David L. Akin
Title: Assistant Professor of Aeronautics and Astronautics

Acknowledgements

First and foremost, I would like to thank my thesis committee, Dr. David L. Akin, Rene Miller, and Dr. Robert Simpson, who have given me guidance and support throughout my work on this thesis. Rene Miller initially brought me into the Space Systems Laboratory, and Dave Akin has been my friend and boss, and I am eternally grateful that he has given me the freedom and latitude to determine which direction of study I would wish to pursue. Dr. Dave has provided me with a flexible work environment which I believe has significantly enhanced my creativity and productivity. Thanks are also due to Dr. Chuck Oman who has provided many valuable comments and suggestions for this thesis.

Throughout my 6 1/2 years of graduate study at M.I.T. (Masters and Ph.D.), there have been many who I am proud to be able to call friends. At the beginning, there were the Preman sisters, Randi and Iris, who I met waiting in line for dinner at Fridays. They made me part of their family, for which I am forever indebted. Now Randi has Myron and two beautiful children, Amanda and Jeremy, who will no doubt bring them all the happiness they truly deserve.

The numerous roommates I have had over the years have been a continuing source of amusement and bemusement (and in some cases headaches). My first roommate, G. Cal Vesely, finally took my advice and married Connie last summer. My friends who helped me write the Canterbury Tales were Kapton Craig Carignan, who is now fighting injustice in the Washington D.C. area, Steve Taylor, perhaps the most brilliant person I have ever met, and my humble Master Ashok Nimgade, whose exploits are legendary. Ashok is perhaps the only person who has visited (and probably knows the entire student body at) every women's college in New England.

At Granada Highlands, I roomed with Steve Fessler, my friend since third grade. When Brian Day moved in, Steve lived in the living room for a year, an experience which none of us will probably ever forget. Brian and Steve were followed by Steve Lackie, who remains today one of my closest friends (even if he did steal my mixed doubles partner). After Steve came The Criminal, who will no doubt get everything she deserves (i.e., a long jail term) for making six months of my life an utter nightmare.

The many people who have made up "The Gang" at Granada are too numerous to name, but some deserve special mention. Herbert Schwarz came from Germany for two years, and was one of the finest friends I have ever had. Then there is my nipper pal Michelle, who taught me how to say "no saaahh" just like a north shore native. Dr. Mark Lee, who accompanied me on my Club Med vacation, has always been a true friend. Then there is my buddy, the beautiful Dee Nelson, who gave me many lessons in giving tennis lessons, and my tennis nemises, including Ian Arvin, Arthur and Todd Giaquinto, Bob and Martha Fitzer, Paul Feng, Rick Mehlinger, Frank McFarland, Warren Epstein, John Lepore, Rob "Max Headroom" Reinecker, and of course Ric Chang, who I met in the singles tournament four consecutive years. I got the last word.

I am eternally gratefully to the many lovely ladies who have given me backrubs over the years: Lynnette, Marguerite, Sharon, Missy, Anna, and Ronni. They have made the grind much easier to bear. I am especially indebted to Veronica Arthur, who has made me smile so many times this past year. She has been much better to me than I have deserved.

Back at the Space Systems Laboratory, there have been many who have brightened up my days. Mary Bowden, John Spofford, and Russ Howard remain behind at the SSL. They comprise the last of the original crew who were there when I began. I wish them a speedy completion to their work. Russ has been a true and faithful friend; we have been through many a Grand Prix race together. To name just a few others, there is Dan "Blackjack" Heimerdinger, who has been a true experience, "Big" Dan Cousins, who has been an all around good guy, Janice Tarrant, my Australian mate, Teresa Villarreal, who has been an excellent skiing companion, and Ping Lee, the labs administrative assistant, who has always had a helpful smile in the midst of all the insanity. It has been a privilege to be a colleague of Tony Marra, who started out working for me when he was a young Freshman. Tony, who has become such an exceptional adult, clearly demonstrates how the student can transcend the master.

On a miscellaneous note, I cannot forget my friends Greg Chamitoff, Bertha "Nikki" Chong, Habib Rahman, and Naomi Guth, who were there to share many good times.

Of course, my former UROPer Edie Erlanson deserves a paragraph of her own. She is a unique individual whose joy for life is a wonder to behold. Edie could always cheer me up in my worst moods, and was (and still is) one of the best friends a person could ever hope for. I am also indebted to Edie's boyfriends, Al and Simon, for being so understanding whenever Edie gave me one of her terrific backrubs. And that Edie's lovely sisters, Ruthie and Bethie, turned out so nice too surely proves that Mom and Dad Erlanson knew what they were doing.

I am indebted to M.I.T. and NASA for providing an exceptional educational experience, and for giving me a Research Assistantship, which payed my tuition and even some of my living expenses over the past 6 1/2 years. Without this support, getting my degree would not have been an option. Many thanks to Roger Cliff, formerly of NASA Goddard, who obtained the initial funds for me to begin this thesis work. The work in this thesis was started under NASA Grant NAG5-445 and continued under NASA Contract NAGW-21.

The name MFIVE (the computer system described in this thesis) comes from the Star Trek episode "The Ultimate Computer" (©Paramount Pictures Corporation) where the computer M-5 ("M" stands for Multitronics, "5" for fifth in the series) was created to "correlate all computer activity of a starship . . . to provide the ultimate in vessel operation and control." Among M-5's functions was to assign crewmembers for landing party duties.

Last, but certainly not least, thank you Mom and Dad, for your support and patience. I think you never really believed I would finish. I love you.

Table of Contents

Page Number

Page Number

Page Number

Page Number

Section 1: Introduction

1.1 General Overview

Past spaceflight experience has shown that current methodologies for performing crew activity scheduling will not be sufficient to support future space station requirements. Current practices for forming Space Shuttle schedules are highly man-hour intensive, and are not amenable to rapid dynamic replanning in the presence of unanticipated or unforeseen events. There is also no way to include the preferences of the crewmembers (for example, the tasks they wish to perform, and when they wish to perform them) into the scheduling process. Further, presently used methods do not incorporate the capabilities of mathematical operations research techniques to perform optimization of schedules in order to maximize the completion of mission goals and find tradeoffs among factors such as differing crewmember skill levels. Scheduling problems have frequently plagued the Space Shuttle, which only operates for periods of a week at a time; on a continuously manned space station in the 1990's, current practices will be totally unacceptable.

In order to remedy this situation, a system will be needed to allow space station crewmembers to interact with ground provided mission priorities and plan out their own schedules. Key to this type of system will be providing crewmembers with a sophisticated and "user friendly" interface which allows them to access a model of the space station work environment. Also needed is the development of mathematical algorithms which will allow a computer to find good work schedules from a user provided database. The framework for such a scheduling system could clearly be extended beyond space station scheduling, to applications such as dynamic mission replanning aboard the Space Shuttle, Space Lab mission planning, and even areas such as airline operations management.

Scheduling theory addresses the problem of the optimal allocation of processors (such as crewmembers) to jobs (such as experiments) subject to a set of constraints (such as time and resource limits). The development of techniques to solve scheduling problems has historically centered around the investigation of idealized scheduling models which often are much simpler than the problems encountered in the real world. Except for the simplest models, presently known mathematical techniques for finding optimal solutions are inadequate for solving large problems.

Several strategies are available for attempting to produce an optimum schedule. In theory, the problem can be cast as an integer programming problem, and solved by any of the various exact techniques available, which are discussed in Section 4.3. However, as explained in Section 4.1 this problem is "NP-hard": the difficulty in finding optimal schedules grows exponentially as the size of the problem grows linearly. Even for problems of modest size, it is therefore much too difficult to find an optimal solution.

In practical applications, heuristic techniques, which do not become exponentially more difficult at the problem grows in size, are often used to solve problems which are otherwise intractable due to exponential growth. Heuristics are rules which can be used to produce solutions of (hopefully) good quality, but which are not guaranteed to be optimal. In some cases, heuristic techniques can be shown to produce solutions which have desirable properties, such as guaranteeing always to be within a certain percent of the optimal solution. For more complicated models, however, even these guarantees may not be possible. In the case of space station crew activity scheduling, the optimization criteria are somewhat inexact (as is, to some extent, the database on which the scheduler relies); hence a heuristic might be considered successful if it could be applied to a set of test problems and shown to consistently produce schedules which are close to optimal. Additionally, having a heuristic which can be compute a solution on a time scale fast enough so that the solution can be used immediately (as would be necessary when performing dynamic reprogramming) is superior to producing a nominally better solution which cannot be obtained in real time. With confidence in such a heuristic, it could then be applied to larger and more complicated problems for which finding an optimum is not a realistic option.

Traditional heuristic methodologies suffer from two basic problems. Firstly, most of the heuristic techniques that have been proposed to solve this type of problem are somewhat inflexible: they rely on a fixed set of rules to produce a single candidate schedule. In practice, it is sometimes found that in "excellent" schedules the jobs will fit together like pieces of a jigsaw puzzle. While a heuristic may produce a candidate schedule of "good" quality (compared to, say, schedules produced by some random sequencing technique), the heuristic may miss the fact that a small change in the schedule might improve results significantly. Further, in complicated scenarios, a technique which is inflexible may not even be able to find feasible, much less optimal, solutions.

The second problem with traditional heuristic methodologies is that they are often unable to adequately address the full range of problems which they must solve. For example, consider a scheduling problem which contains numerous time constraints on the jobs to be scheduled. A heuristic strategy which works well for finding good schedules heavily dominated by time constraints might fail miserably on schedules dominated by competition for scarce resources.

In order to bridge these difficulties, this thesis develops an iterative scheduling technique which attempts to search (span) the combinatoric solution space in an intelligent manner, much as many methods (such as gradient search) do for continuous domain problems. The technique examines (and makes perturbations upon) successive schedules in an attempt to find progressively better solutions; it thus avoids becoming trapped into some fixed scheduling. The technique is also flexible enough to accomodate highly complicated constraint environments; the search inherently focuses upon the parts of a schedule which make it complicated, while avoiding dealing with factors which are not present in a particular problem instance.

In dealing with heuristics, it is often the case that it is difficult, if not impossible, to analytically prove that the techniques which tend to perform best in practice are "robust." The technique developed in this thesis is no exception. In order to validate the quality of the model, it is therefore necessary to test the model via monte carlo simulations over realistic scenarios.

This thesis therefore applies this iterative technique to the space station crew activity scheduling model, which is so complex that previous heuristic methodologies have proven largely inadequate. Without a good heuristic, past solution strategies have had to either implement poor solutions or rely upon a person exercising "professional judgment" in order to derive a good schedule. While there is nothing wrong with these schedules (i.e., schedules formed by strategies which are not sufficiently understood by the human scheduler to make them into an algorithm), producing them is highly man-hour intensive, non-reproducible, and not amenable to automated processing. It is shown that the iterative technique developed herein is capable of quickly generating high quality schedules for this problem.

1.2 Organization of this Thesis

This thesis is divided into six sections. In Section 2, the motivating reasons are given for performing this research and for building a computer based tool for aiding in the performance of crew activity scheduling.

Section 3 describes the crew activity planning problem, and discusses how scheduling functions should be divided between space based crewmembers and ground personnel. This section also presents an historical overview of relevant operations research and artificial intelligence techniques, and discusses other computer based tools which have been developed to deal with problems similar to the crew activity planning problem described in this thesis. Finally, a detailed mathematical model of the crew activity planning problem is presented.

Section 4 reviews the available literature and presents the theoretical framework developed for this thesis. In Section 4.1 the computational complexity of the crew activity planning problem is discussed, and a proof is given that the problem is NP-Hard. Section 4.2 presents a methodology developed to simplify the scheduling problem by using inferences to propagate interrelated time constraints. Section 4.3 discusses previous research into the resource constrained multi-project scheduling problem, a problem which is encompassed in the more general crew activity planning problem. Also discussed in this section are several methodologies that have been developed in studies which specifically dealt with the crew activity planning problem. Section 4.4 presents several approaches which were examined (with mixed success) for searching the solution space for good solutions, including the iterative solution technique which was adopted and used in Section 5.

Section 5 presents the details of the operation of the iterative algorithms which were implemented, including a new heuristic technique, called the Maximum Compatibility Method, which proved extremely efficient at finding solutions to certain examples of the crew activity planning problem. Section 5 also presents the results of applying many different versions of the iterative algorithm to a variety of problems. The complexity of the iterative algorithm is discussed, and guidelines are given for using the iterative algorithms to solve scheduling problems.

Section 6 demonstrates the operation of the MFIVE computer scheduling tool which was developed to maintain information about space station equipment, personnel and activities. MFIVE demonstrates the ability to perform interactive scheduling and to test the algorithms developed in this thesis. Finally, general conclusions are presented in Section 7, as well as recommendations for further research.

Section 2: Motivation

Activities onboard a space station may be grouped into two categories: core activities (those required to keep the station operating and the crew in good health) and mission activities (those relating to performing the jobs for which the station is required, such as satellite servicing, EVA structural assembly, materials processing and manufacturing, and technology experiments). While both types of activities are necessary, they may be thought of as competing for the limited resources of the space station crew. Any savings in time or effort either in the completion of core activities or the efficiency of performing mission activities would then become available to perform additional mission-oriented activities. Skylab showed that current methodologies used for the planning of crew activities requires a sizable work force at JSC Mission Control; a similar situation has also developed for Space Shuttle operations.

There is also evidence that the psychological health of a space station crew would be enhanced if they could be given some control over their schedules, as would be provided by an interactive scheduling system. Scheduling done from the the ground, without crewmember input, has often resulted in "overcontrolling" of the crew, with negative consequences. An example of this occurred during the last Skylab mission in December 1973. According to the flight director at the time:

> "We send up about six feet of instructions to the astronauts' teleprinter every day, at least 42 separate instructions telling them where to point the solar telescope and which scientific instruments to use. We lay out the whole day for them, and they normally follow it to a T. We've learned how to maximize what you can get out of a man in one day" [Balbaky, 1980].

After over a month in space, the three astronauts conducted the first "strike in space" by suspending ground communication and taking a day off [H.S.F. Cooper, 1976].

It has therefore been suggested that the utility and autonomy of space station operations could be greatly enhanced by the incorporation of computer systems utilizing expert decision-making capabilities and a relational data base. An expert decision-making capability will capture the expertise of many experts on various aspects of space station operations, for subsequent use by nonexperts (i.e., spacecraft crewmembers). Key features of such a system would be its ability to explain, on request, how it arrived at its decisions, and the capability for users to reject the system's conclusions if they disagree with them. The information utilized would be stored in a relational data base that would be used by the system in response to a current problem: the expert system uses rules to generate conclusions based on the information stored in the relatational data base. It is envisioned that a unified space station computer system and database would be able to support a wide variety of functions, such as those listed in Table 2.1.

During the summer of 1983, NASA sponsored a summer workshop which investigated the potential ways in which machine technology could potentially affect and augment space station operation. One of the conclusions of that study, in which the author of this thesis was a participant, was that crew activity planning was an attractive area for space station automation [Johnson, et al., 1985]. This thesis will therefore focus upon demonstrating the feasibility of onboard crew activity planning. This demonstration includes the design and implementation of a prototype computer system to show the feasibility of onboard crew activity planning [a users guide appears in Appendix B, and a programmers guide appears in Appendix C], and the development of algorithms to assist the crewmember in efficient performance of that function. Although not part of this thesis, a system for locating stowed equipment has been incorporated into the crew activity planning system [Kranzler, 1986]. This "stowage logistics clerk" shares a unified database with the planning system; the use of tools and equipment is associated with the performance of onboard crew activities.

Table 2.1: Potential Space Station Knowledge Base Applications

(not an exhaustive list)

1 Subsystems Management Tasks
- 1.1 Power Subsystem Management
- 1.2 Thermal Subsystem Management
- 1.3 Life Support Subsystem Management
- 1.4 Information Subsystem Management
- 1.5 Propulsion Subsystem Management
- 1.6 Communications Subsystem Management
- 1.7 Fluids Subsystem Management
- 1.8 System Performance Evaluations/Trends Analysis
- 1.9 Consumable Inventory and Resupply
- 1.10 Fault Detection and Annunciation
- 1.11 Troubleshooting/Malfunction Procedures
- 1.12 In-Flight Maintenance

2 Crew Activity Planning
- 2.1 Daily Housekeeping Chores
  - 2.1.1 Trash Removal
  - 2.1.2 General Cleaning
  - etc.
- 2.2 Maintenance and Repair Schedules
- 2.3 Crewmember Health Maintenance
- 2.4 Construction Activities
- 2.5 Satellite Servicing Activities
- 2.6 Scientific Experimentation Activities
- 2.7 Scientific Observation Activities
- 2.8 Training Activities
- 2.9 Crew Rotation

3 Trajectory/Flight Dynamics
- 3.1 Space Station Orbital Maintenance
- 3.2 Orbiter Transfer Vehicle Tracking and Maneuver Planning
- 3.3 Satellite Rendezvous Tracking and Maneuver Planning
- 3.4 Orbiter Rendezvous Tracking and Maneuver Planning

4 Construction/Satellite Servicing
- 4.1 Assemble Structures/Spacecraft
- 4.2 Satellite Retrieval/Servicing
- 4.3 Satellite Checkout

5 Long Term Planning/Logistics

Section 3: Crew Activity Planning

3.1 Division of Scheduling Functions Between Space and Ground Crews

The entire process of planning crewmember activities is a highly complex one, involving consideration of and tradeoffs among crewmember workloads, resource usage, job priorities, and time and target constraints of various jobs to be performed by the crew. For shuttle missions, all planning and scheduling functions are currently done by ground schedulers (known as timeline engineers). Due to the short nature of a shuttle mission and the very limited amounts of in-space crew time, it will probably be necessary for a significant portion of shuttle mission planning to remain ground-based, but on-orbit scheduling activities could still enable functions such as replanning in the event of unanticipated deviations from the nominal schedule. Current practices dictate a "return to baseline" after any unforseen event, but a greater achievement of mission priorities and much efficiency could be gained by allowing a more complete rescheduling of mission activities. Additionally, crew confidence and moral would be enhanced by allowing them the ability to examine previously unanticipated mission timelines.

Aboard a space station, where people will be living for months at a time, it would be highly desirable for the crew to perform as much of the planning and scheduling process as is feasible. Due to the long duration of space station stays, it will be necessary to organize rational work and rest schedules which maintain crewmember health. Work cycles based on a 24 hour day may be standard, but allowance must be made for the occurrence of irregular or emergency situations which would require other work schedules [Litsov and Bulyko, 1983; Ivakhnov, 1984]. Requiring the crew to entirely and autonomously manage the entire scheduling process, however, would require them to deal with vast amounts of data and perform functions which are beyond their areas of expertise; this would consume large amounts of valuable crewmember time. Timeline engineer input will still be necessary to assist the space crew, but in a much reduced role (both in terms of scope and manpower) over that necessary without a space based crew activity planning system. Timeline engineers will still have mission responsibility giving them a strong need to be involved, as well as a global knowledge of mission specific details, some of which might not be included in any computer based model. Table 3.1 indicates a functional allocation scheme for allocating crew scheduling activities between ground controllers and the space based crew.

Scheduling Activity or Task	Location of Activity or Task		
	Ground Operations	Expert System	Crew
Design of Expert System	✓		
Test and Demonstration of Expert System	✓		✓
Maintenance and Adaptation of Expert System	✓	✓ (long term)	✓
Assemble Input Data: Time Estimates, Resource Requirements, Constraints	✓		
Input Daily Task Priorities and Update Information	✓		✓ (secondary)
Preliminary Timeline Computation (Days in Advance)	✓ (monitoring)	✓	
Compute/Review Next-Day Timelines (Overnight)	✓ (monitoring)	✓	✓ (interactive)
Review/Update/ Recompute Morning Timelines	✓ (monitoring)	✓	✓ (interactive)
Approve/Accept Final Morning Timelines	✓		✓
Graphically Represent Timelines		✓	
Recommend Deviations from Daily Timelines	✓	✓	✓
Contingent Recomputation of Daily Timelines		✓	✓

Table 3.1: Functional Allocation of Crew Scheduling Activities

Ground controllers must retain the ability to provide input to any space based activity planning system, but generally not in terms of short-term real-time control. Ground control will necessarily have strong input to overall mission priorities, and will need to provide the detailed technical data that will serve as computational parameters regarding requirements, resources, and constraints. This data will include such items as space station orbital parameters (e.g. inclination, apogee, perigee), job descriptions, time windows for performing the job, and instructions to assist the crew in performing the job. Ground operations will also undertake preliminary trial scheduling runs to help establish general mission feasibility and priorities.

Onboard a space station, the scheduling system could then be operated in an interactive mode on a daily (or other routine) basis to determine crew work timelines. At the beginning of the work day, the crewmember in charge of scheduling functions could use the scheduling system to determine work timelines for that day. Activities could be interactively scheduled into time slots and assigned to crewmembers, with the scheduling system simply pointing out existing windows which satisfy all resource requirements and time constraints. This mode of operation affords the greatest flexibility to crewmembers in determining their work schedules. Subjective preferences by the crew may have minor effects on schedule quality, but large effects on crew morale. However, if all the activities which the crewmembers are to perform are scheduled in this manner, it would place an unacceptable burden on the crewmember performing the scheduling. In addition, it would not allow the utilization of mathematical techniques for scheduling optimization, which would maximize use of crewmember capabilities, utilization of the limited resources available, and achievement of mission goals.

A second mode of scheduler operation would allow the program to automatically plan out the remaining jobs. The scheduler would couple operations research algorithms and heuristics with the large computational capabilities of computers to generate optimal (or near optimal) timelines. Schedules could thus be determined in minutes which would have efficiencies comparable to ones requiring weeks or months of manpower to generate using traditional dispatching or trial and error approaches.

The scheduler would then provide crewmembers with printouts of timelines, instructions for performing the various jobs, and the locations of various tools. At the end of the work day, the scheduler could be briefed on the success or failure of each of the various jobs done that day (e.g., which items had to be rescheduled, which were not performed due to lack of time, etc.). If needed, the system could also be consulted during the work day for additional information and replanning.

A high priority in the development of a scheduler is to keep its operation as simple as possible, and to require as little crewmember training as possible to operate, while still permiting a large amount of crewmember input if desired. If the system is so cumbersome that it must be used constantly or require highly specialized training and skills to operate effectively, then the crewmembers will probably not want to use it at all.

3.2 An Overview of Planning and Scheduling: Artificial Intelligence Techniques

It is only within the past twenty years that artificial intelligence techniques have been available to assist in task planning [Gevarter, 1982]. Artificial intelligence research has contributed in three areas relevant to crew activity planning: expert systems, which code human expertise in a particular problem domain into the form of rules, which can then be used to enable a computer to operate with the same expertise as a human expert; planning systems, which can use spatial and temporal reasoning to plan actions in order to achieve desired goals; and neural networks which combine a parallel architecture with nonlinear analog signal processing to rapidly find solutions to combinatorial optimization problems [Hopfield and Tank, 1985; Hopfield and Tank, 1987].

NOAH was an early planner which dealt with interacting subgoals. The method of least commitment and backward chaining initially produced a partial ordering of steps for each plan. When interference between subgoal plans was observed, the planner adjusted the ordering of the steps to resolve the interference and produce a final parallel plan with time ordered steps. Other planning systems include STRIPS, which was developed to perform such tasks as stacking blocks to achieve a particular goal formation, and MOLGEN, which was designed to plan experiments in molecular genetics. The NUDGE system was developed to understand incomplete and possibly inconsistent management-scheduling requests, and to provide a complete specification for a conventional scheduling algorithm. Perhaps one of the projects most closely related to crew activity planning is the ISIS expert system developed at Carnegie-Mellon University to perform job shop scheduling functions [Smith and Ow, 1985; Fox, et al., 1983; Smith, 1983; Townsend, 1983]. The OPAL expert system [Bensana, et al., 1986] and the MASCOT expert system [Erschler and Esquirol, 1986] are being developed in France for aiding in job shop scheduling.

Work at Ford Aerospace has been investigating the use of expert systems to automate spacecraft ground command and control functions [Wagner, 1983]. Boeing has been interested in expert systems for cruise missile automated mission planning [Jardine and Shebs, 1983] and for space station operations [Stein, et al., 1986] including environmental, life support, and power systems [Marsh, 1984b]. Contel SPACECOM has developed the ESSOC expert system [Rook and Odubiyi, undated] for aiding in satellite control operations.

Over the past several years there has been great interest within NASA for using expert systems for space applications. The NAVEX expert system has been developed at the NASA Johnson Space Center to perform navigation functions during Space Shuttle reentry [Marsh, 1984a; Marsh 1984b]. Systems also exist for performing shuttle electrical system checks during prelaunch ground preparations [Marsh, 1984b]. MITRE, under contract to the Kennedy Space Center, is developing the EMPRESS system [Hankins, et al., 1986] to assist in the planning of processing activities for cargo manifested to fly on the Space Shuttle.

One of the first planning systems which NASA developed was the Deviser system [Vere, 1983a; Vere, 1983b; Vere, 1985], built at JPL to autonomously plan a spacecraft's actions during a planetary flyby. Deviser was developed from NOAH, but unlike NOAH, it is capable of handling goals with time constraints and durations. JPL is also developing a system called PLAN-IT [Grenander, 1985], an expert system for schedule planning. Like Deviser, PLAN-IT is written in Lisp on a Symbolics 3600 computer. The Deviser system figures out and plans the steps necessary to perform an action, whereas PLAN-IT decides when to schedule the actions. It is anticipated that PLAN-IT will be tested on Spacelab scheduling.

The AMPASES expert system [Jakubowicz, 1985], commissioned by NASA Goddard Space Flight Center, is under development on a Symbolics 3670 computer by GE and TRW for both interactive and automated space station mission planning. The MITRE Corporation has developed a Symbolics 3600 based expert system called KNEECAP [Mogilensky, et al., 1983], also commissioned by Goddard, for crew activity planning aboard the space shuttle. KNEECAP is a frame-based system derived from the architecture used by MITRE for their KNOBS expert system [Engleman, et al., 1983; Scarl, undated], which was constructed to aid in support of Air Force tactical air mission planning. KNEECAP was intended to primarily provide an interactive scheduling capability.

The Space Station Experiment Scheduler (SSES) [Touchton, undated] has been developed by Technology Applications, Inc., under contract to the NASA Marshall Space Flight Center. SSES is implemented in Golden Common Lisp on an IBM-PC/AT. SSES is designed to schedule experiments and payloads according to priority, time, resource, and crew constraints. SSES also has the capability to dynamically reschedule a portion of a mission if constraints are modified.

The NASA Ames Research Center has developed a system called OpSim which runs on an Apple Macintosh computer. OpSim is based on commercially available file management software, and models Space Station operations, such as crew activities and equipment usage [NASA, 1986].

A crew activity planning system does not in general entail many of the problems encountered by planning expert systems such as the Deviser, STRIPS, or MOLGEN. In such systems, it is necessary to generate, using a set of operators, each of the steps necessary to achieve a goal, and to then break the planning task into a hierarchy of subgoals. In the crew activity planning problem, however, each of the steps is given, and the task is to arrange these steps (in time) so that they do not violate any of the constraints (e.g., crew, time, and resource constraints) which are imposed upon them. These constraints are often much more complicated than those encountered by block stacking programs (such as STRIPS), and the most difficult task is often to mutually satisfy these constraints.

On the other hand, several other difficulties faced by other planners are shared by a crew activity planner. If a situation is very complicated, the planner must be able to focus on the most important considerations. Additionally, in a scenario where there are more tasks to (ideally) perform than are physically possible, then the planner must be able to prioritize the tasks to be scheduled. Interactions between operations to be scheduled often occur, and the planner must be able to recognize these interactions and cope with them. Further, often the planning context will only be approximately known (e.g., some of the parameters may be random variables), so that the planner must operate in the face of uncertainty. This requires preparing for contingencies. Finally, if a plan is to be carried out by several people, activities may require the simultaneous attention and cooperation of several crewmembers [Hayes-Roth, et al., 1983].

3.3 An Overview of Planning and Scheduling: Operation Research Techniques

Whereas the field of artificial intelligence contributes techniques to solving scheduling problems which attempt to model intelligence, operations research addresses the scheduling problem from a largely mathematical point of view. The branch of operations research known as deterministic scheduling theory encompasses mathematical techniques for finding an optimal scheduling of a number of jobs among several processors (in this case, crewmembers). This field dates back to the 1950's, but has been the subject of intensive investigation over the past several years [Conway, et al., 1967; Graham, et al., 1979; Dempster, et al., 1982]. A good overview of the field is provided in the survey paper by Lawler, Lenstra, and Rinnooy Kan [Lawler, et al., 1981]; scheduling subject to resource constraints is surveyed in [Blazewicz, et al., 1980]; and scheduling subject to precedence constraints is surveyed in [Lawler and Lenstra, 1981].

NASA has performed several studies involving space applications for scheduling theory. The Crew Activity Scheduling Program (CASP) was developed to aid in scheduling of Apollo missions [Murphy, et al., 1968]. Fisher and Jaikumar developed an algorithm for the scheduling of Space Shuttle launches. The algorithm selected mission launch times that minimize the number of missions flown late, and satisfy early start time and resource constriants [Fisher and Jaikumar, 1978]. A number of systems have been developed at NASA for scheduling, including the Fast Automated Scheduling Technique system (FAST), the Marshall Interactive Planning System (MIPS), the Manned Activity Scheduling System (MASS), the Viking Lander Sequence of Events Scheduler (LSEQ), and the Crew Activity Planner (CAP) [Hitz, 1976].

3.3.1 The Crew Activity Planning Model

A detailed inspection of a prototypical space station flight plan illustrates the complexities involved in producing a schedule. Typically, a flight plan consists of a set of jobs (or tasks), ($j = 1, \ldots, n$) (usually on the order of hundreds) to be performed by crewmembers ($i = 1, \ldots, m$) (typically eight or less) over a period ranging from several hours to several months. In the case of long range planning, jobs may be higher level goals, and the planning window may be on the order of weeks to years. In general, each job has a default time for completion (p_j), although in some cases different crewmembers may have different skill levels and thus different completion times (p_{ij}, $p_{ij} \geq 0$ for all i,j). Further, in many cases only some of the crewmembers will be rated for a specific job, and the remaining crew is effectively incapable of performing that job ($p_{ij}=\infty$).

In a proposed schedule, let $x_{ij} = 1$ if Crewmember i is assigned to Job j, and $x_{ij} = 0$ otherwise. Let s_j denote the start time of Job j.

In general, the scheduler must search for solutions (for the s_j and x_{ij}) in an environment which contains numerous types of constraints. The following is a list of typical constraints involved in space station crew activity planning (unless otherwise noted, each of these constraints are implemented in the scheduling problems discussed in Section 5 and in the MFIVE scheduler discussed in Section 6):

1) Each job requires a specific number of crewmembers to perform it (i.e., parallel, unrelated processors) (c_j) ranging from zero to the total number of crewmembers available on the space station ($c_j = \sum_i x_{ij}$). A job might not require any crew if, for example, it is a necessary processing or waiting step in a long sequence of interlinked jobs. By convention adopted in this thesis, <u>if a job involves multiple crewmembers, they must all begin the job at the same time, s_j</u>. If the different crewmembers have different processing times for the job, then it is possible that they will finish at different times. In this case, any resource usage by the job (see 8, below) will extend until all the crewmembers have completed the job.

2) Each flight plan has a start time (START), before which no job can be planned, and an end time (END), after which no job may be performed. ($s_j \geq$ START for all j, $s_j + \max p_{ij}x_{ij} \leq$ END for all j)

3) Each job may have an earliest start time (release date) (EST), latest (LST), or required start time (RST) and a latest end time (due date) (LET). (i.e., $s_j \geq EST_j$, $s_j \leq LST_j$, $s_j = RST_j$, $s_j + p_{ij}x_{ij} \leq LET_j$ for all i)

4) Precedence relations may exist between jobs (j,k), which stipulate that one job must be completed before a second job is begun. (i.e., $s_j+p_{ij}x_{ij} \leq s_k$ for all i)

5) Concurrence relations may exist between jobs, which stipulate that the jobs must begin at the same time. (i.e., $s_j = s_k$)

6) Time constraints may exist between jobs which designate minimum and maximum intervals between the start of the jobs. (i.e., $s_j + \Delta MIN_{jk} \leq s_k$, $s_j + \Delta MAX_{jk} \leq s_k$)

7) Preemption (job splitting) is not allowed: the processing of any job cannot be interrupted and resumed at a later time. (In fact, in real world operations, the preemption of a job may be feasible at certain times. Historically, formulations of this problem have avoided this issue primarily because of the complexities involved in specifing times or time periods in which preemption is allowed, and in enabling an algorithm to make sensible use of this information. In practice, if one wishes to allow preemption at certain times, then the job is simply separated into several smaller jobs linked by precedence constraints.)

8) Renewable resource constraints may apply to some jobs. For example, there may be a limit (possibly time varying) on the total amount of power consumed by all jobs occurring at any one time. Resource usage by individual jobs may also be time varying. Maximum resource levels may even be a function of when other jobs occur in the schedule. This could occur if some job produced additional amounts of a resource (e.g., power) or if some job drained reserve supplies which might not be replenished until some later time. (Let $r_{jz}(\Delta t)$ be the amount of Resource z used by Job j at time Δt after the start of the job ($r_{jz} = 0$ for $\Delta t < 0$). Let $R_z(t)$ be the maximum amount of Resource z available at time t. Then for a renewable resource, $\sum_j r_{jz}(t - s_j) \leq R_z(t)$ for all z, and for all t such that START $\leq t \leq$ END. More generally, R_z may be a function of the s_j's: $R_z = R_z(t, s_1, \ldots, s_n)$.) Note: in the implementation of the MFIVE scheduler used in this thesis, neither resource limits or resource usages are allowed to be time varying.

9) Nonlinear resource constraints are possible. (For example, some job which is audio or vibration sensitive might only be scheduled while there are no other jobs scheduled which produce noise or vibration. For each Job j, a quantity $h_j(t)$ can then be defined such that $h_j(t) = 1$ during the performance of the job if it is noise or vibration sensitive, $h_j(t) = -1$ during the performance of the job if it is noise or vibration generating, and $h_j(t) = 0$ for all other jobs and times. Then it is sufficient that $(\sum_j h_j(t)^2)^2 = (\sum_j h_j(t))^2$ for all t such that START $\leq t \leq$ END.)

10) Non-renewable resource constraints might also extend to total usage over some time interval (e.g., limiting the total amount of food which could be consumed during any one day). (Let R_{zg} denote the amount of Resource z available in period g, and $r_{jz}(\Delta t)$ denote the rate of usage of Resource z by Job j at time Δt from the start of Job j. Then $\int_g r_{jz}(t - s_j)dt \leq R_{zg}$.) Note: in the implementation of the MFIVE scheduler used in this thesis, there is no capability to specify non-renewable resources.

11) A job may have multiple target time windows during which it can occur, and these windows may be given by the solution (either exact, numerical, or logical) of arbitrary functions of time. For example, it may be required that space station orbital parameters be within certain ranges in order to perform a specific job. The applicable equations can then be solved (either exactly, through some iterative search procedure, or through table lookup) to give time windows for the job.

In addition to these above constraints, other types of requirements can be specified which will facilitate the scheduling process. While each of these requirements is decomposable into constraints of the type already discussed, they offer conceptual advantages to specifying each of those constraints separately. For example, it might be advantageous to directly specify multiple performances of a job, with maximum and minimum time intervals between the completion time of one performance and the start time of the next performance (e.g., sleep periods could be scheduled on a daily basis with an interval of 15 to 17 hours between the end of one sleep period and the start of the next). Other constraints could allow jobs to be grouped together into larger units, with maximum and minimum time intervals between various jobs and a maximum duration on the performance of the entire unit. Note: none of these types of requirements are implemented in the version of the MFIVE scheduler used in this thesis. Of course, even though it is more cumbersome, the same effect can still be achieved by using the types of constraints which are implemented (e.g., precedence constraints, ΔMIN, and ΔMAX constraints).

A schedule is said to be complete if it successfully schedules all jobs within their constraints. A schedule is said to be feasible even if it does not schedule all jobs, as long as it does not explicitly violate any constraints. A schedule is termed optimal if it is feasible and minimizes (or maximizes) the desired objective function. A schedule need not be complete to be optimal.

The goal of the scheduler is then to produce a schedule which maximizes the weights, w_j, (assigned by some user provided prioritization scheme) of the jobs successfully scheduled (there may be more jobs to perform than can be fit into the timeline), equalizes individual workload, minimizes total workload, and meets subjective approval of the crewmembers. A schedule which is optimal by one of these criteria may not be optimal by the other criteria, and hence a schedule should employ an algorithm which produces a "reasonable" combination of all the above criteria. (It should be noted that if the objective is to maximize the weights of the jobs successfully scheduled, then any schedule which is complete will also be optimal.)

While the above model is quite general in terms of accomodating a broad spectrum of constraints, there are some types of scheduling problems which are not addressed by this model. For example, while the duration of an activity may be a function of the crewmember(s) assigned to it, for any given crewmember the performance time is always considered a fixed quantity. Some of the scheduling literature addresses problems where job processing times may be a function of the resources allocated to them, or where job weights (w_j) may be a function of the amount of processing time devoted to the jobs. These variants of the scheduling problem are not covered by the above model.

Section 4: Theoretical Discussion of Solution Techniques

4.1 Computational Complexity

The difficulty of a combinatorial optimization problem (such as the crew activity planning problem) is usually characterized by the efficiency of the best known algorithm for solving the problem optimally. An algorithm is said to be efficient (or polynomial-time) if its execution time grows as a polynomial in the size of the input. Conversely, an algorithm is not considered efficient if it grows exponentially in the size of the input. It should be noted, from a practical standpoint, that this notion of efficiency can be somewhat misleading. It may be the case that an exponential algorithm may outperform a polynomial algorithm for practically sized problems.

A problem is said to be in the class *NP* if there exists a polynomial time algorithm for checking that a proposed solution to the problem is correct. The class NP includes problems that can be solved with polynomial time algorithms as well as problems for which no polynomial solution algorithm is known. Some of these problems, for which there are no known polynomial algorithms, fall into a subclass called NP-complete problems. An additional characterization of NP-complete problems is that if a polynomial algorithm exists for any NP-complete problem, then there are polynomial algorithms for all NP-complete problems [Papadimitriou and Steiglitz, 1982]. In spite of much research, no polynomial solution for any NP-complete problem has been found, and it is widely conjectured that none exists. On the other hand, no one has been able to prove that no polynomial solution exists.

NP-Hard problems are closely related to NP-complete problems. NP-Complete problems are called recognition problems, which means that they always have a yes or a no answer. A typical objective of an NP-complete minimization problem would be to ask if there is a problem solution of less than a given value. Once given a candidate solution, one could easily check whether or not it is indeed a valid solution to the stated problem, and whether or not it has the asserted value.

The NP-Hard version of the same problem would instead ask for the lowest valued solution to the minimization problem. If given a candidate optimal solution, one could easily check that it is a valid solution and that it has an asserted value, but there is no way to show that it is, in fact, the minimum such solution. The NP-Hard version of an NP-Complete problem is therefore considered to be at least as hard as the NP-Complete problem itself.

There are three principle methods for showing that a problem in NP is NP-complete [Papadimitriou and Steiglitz, 1982]. The first is to show that all other problems in NP polynomially transform to the problem. The second method is to show that there is a polynomial time transformation from some other problem in NP which has already been shown to be NP-complete. The third method is to find some special case of the problem which can be shown to be NP-complete. The third method will be utilized to show that the recognition version of the crew activity planning problem is NP-complete, and the problem of finding an optimum solution is therefore NP-hard.

The problem of finding a complete schedule for the general crew activity planning problem is NP-complete. Clearly it is in the class NP, because any proposed schedule can be easily checked (i.e. in polynomial time) to ensure that all jobs are included and that no constraints are violated. Consider the special case of the crew activity planning problem where there are only two crewmembers, and no constraints except deadline (latest end time) constraints. Further, assume that both crewmembers have identical completion times, p_j, for each Jobs j = 1, ..., n. Suppose it is the case that all the jobs have the same required latest end time LET_j, which is defined so that $LET_j = 1/2\sum_j p_j$ for all Jobs j. This problem is the same as that of finding a subset S of the j's such that $\sum_{j \in S} p_j = \sum_{j \notin S} p_j$. This is the PARTITION problem, which is known to be NP-complete [Papadimitriou and Steiglitz, 1982].

4.2 Propagating Time Constraints in a Schedule

In this section a methodology is developed for actively propagating generalized time constraints relating jobs to other jobs and fixed points in time. If crewmembers can have differing processing times for the same job, one can only determine *a priori* (before crewmember assignment) a minimum and maximum bound on the duration of each job. An extension to the standard critical path method [Kelly and Walker, 1959] is developed to accommodate this uncertainty in job duration, as well as a richer spectrum of constraints than are usually addressed in critical path problems. It is then shown how to narrow down the size of each job's feasible time window by making maximum possible inference from the constraints.

The time windows thus obtained by this algorithm guarantee that all jobs can be sequenced (i.e., ordered, without assignment to any crewmember) within their planning windows so that they observe all time constraints. As will be shown, the algorithm can further guarantee that if a job is scheduled to start after its earliest start time and before its latest start time, and to end before its latest end time, as determined from constraint propagation, then it cannot possibly force some other job to become infeasible (i.e., become unschedulable within its time constraints). Of course, it is still possible to have an infeasible schedule because of conflicts caused by resource limits, targets, or a finite number of processors (crewmembers), but this algorithm is of much benefit in finding good solutions to the crew activity planning problem, and goes a long way towards reducing the complexities of avoiding infeasibilities.

4.2.1 Problem Overview

Consider the problem of scheduling a set of n jobs. Numerous types of time constraints, as were described in Section 3.2.1, may be specified for each job. Some constraints relate performances to fixed points in time: earliest start time constraints, denoted $EST(x) = \mu$, specify that Job x must begin no earlier than time μ; latest start time constraints, $LST(x) = \mu$, specify that Job x must begin no later than time μ; required start time constraints, $RST(x) = \mu$, specify that Job x must start at time μ; and finally, latest end time constraints, $LET(x) = \mu$, specify that performance x must be completed by time μ. Additionally, every schedule has some minimum time, START, before which no job may be scheduled and some maximum time, END, after which no job may be completed.

Other constraints relate jobs to other jobs. Precedence constraints, $P(x,y) = 1$, specify that Job x must be completed before the start of Job y. Concurrence constraints, $CO(x,y) = 1$, specify that Jobs x and y must begin at the same time. Minimum delta start time constraints, $\Delta MIN(x,y) = \mu$, indicate that there must be at least an interval of length μ between the start of Job x and the start of Job y. Maximum delta start time constraints, $\Delta MAX(x,y) = \mu$, indicate that the maximum interval between the start of Job x and the start of Job y is μ.

For each Job k (k=1, 2, 3, ..., n), we can compute the values MAXT(k) and MINT(k) where MAXT(k) is the maximum time that it could take to perform Job k and MINT(k) is the minimum time that it could take to perform Job k. For example, if k is a job requiring one crewmember, then MINT(k) is just the performance time of the crewmember who can complete the job fastest. In the absence of more complete information, one can always set MINT(k) = 0 and MAXT(k) = infinity (or some other large value, such as END - START).

Propagating the constraints can yield numerous inferences about the feasible time windows for each job. For example, if $P(x,y) = 1$, then the earliest start time for Job y can be inferred to be at least as late as the earliest start time for Job x plus the minimum performance time of Job x, i.e., $EST(y) \geq EST(x) + MINT(x)$. What is desired is an algorithm for making <u>maximum</u> inference from the specified constraints to narrow down the feasible time windows for each job.

4.2.2 Constraint Reduction

The first step in reducing the problem is to add an event * which has zero duration (MINT(*) = MAXT(*) = 0) and required start time START (RST(*) = START).

The second step is to change all constraints to ΔMIN and ΔMAX constraints. This is accomplished by the following transformations:

$EST(x) = \mu$	becomes	$\Delta MIN(*,x) = \mu - START$
$LST(x) = \mu$	becomes	$\Delta MAX(*,x) = \mu - START$
$RST(x) = \mu$	becomes	$\Delta MIN(*,x) = \Delta MAX(*,x) = \mu - START$
$LET(x) = \mu$	becomes	$\Delta MAX(*,x) = (\mu - START) - MINT(x)$
$P(x,y) = 1$	becomes	$\Delta MIN(x,y) = MINT(x)$
$CO(x,y) = 1$	becomes	$\Delta MIN(x,y) = \Delta MAX(x,y) = 0$

(It should be noted that $\Delta MAX(*,x) = (\mu - START) - MINT(x)$ is necessary but not sufficient for $LET(x) = \mu$, and that $\Delta MIN(x,y) = MINT(x)$ is necessary but not sufficient for $P(x,y) = 1$. However, until event x is scheduled, we cannot make any better inference from these constraints than is given above.)

The third step is to change all ΔMIN constraints to ΔMAX constraints. This is done by replacing all $\Delta MIN(x,y) = \mu$ by $\Delta MAX(y,x) = -\mu$. If either this step or step 2 produces more than one value for ΔMAX(x,y), then it is important to keep track of only the smallest value, because it represents the most binding constraint.

Finally, the last (fourth) step is to form the complete ΔMAX matrix. This is accomplished by initializing:

$$\Delta MAX(k,*) = 0 \qquad k = *, 1, 2, \ldots, n$$

$$\Delta MAX(k,k) = 0 \qquad k = 1, 2, \ldots, n$$

$$\Delta MAX(m,n) = (END - START) - MINT(n) \qquad m = *, 1, 2, \ldots, n$$

$$n = 1, 2, \ldots, n$$

$$m \neq n$$

and then substituting each of the ΔMAX values found in steps 1 - 3 into the ΔMAX matrix whenever they are less than the corresponding value already entered in the matrix.

4.2.3 Inference Generation

Time windows for each of the jobs may now be found by applying a shortest path algorithm (such as Floyd's or Dijkstra's) to the ΔMAX data by considering each of the jobs as a node and each of the constraints as an arc on a graph. For example, as shown in Figure 4.1, if $\Delta MAX(*,1) = 5$, $\Delta MAX(1,2) = 8$, and $\Delta MAX(*,2) = 20$, then we can replace ΔMAX(*,2) by 13 because the first two constraints form a tighter limit (shorter path) on the maximum interval between the start of the timeline and the start of Job 2. The shortest path algorithm will completely search the ΔMAX information to find the shortest path by any possible route through the matrix. (A complete worked example of constraint reduction and the shortest path algorithm is provided in Section 4.2.7.)

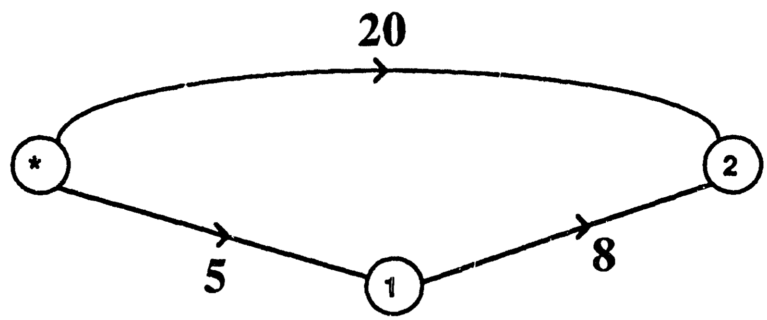


Figure 4.1: Finding a Shortest Path From * to 2 With Constraint Propagation

Let the shortest path algorithm yield the matrix ΔSMAX, containing the length of the shortest path from every node to every other node in the ΔMAX graph. Then the latest start time (as inferred) for each job is then given by LST(x) = START + ΔSMAX(*,x), x = 1, 2, ..., n.

Similarly, we can find the inferred earliest start times for each of the jobs: EST(x) = START - ΔSMAX(x,*), x = 1, 2, ..., n. For example, if ΔSMAX(x,*) = -8, then there is at least 8 time units between the start of the schedule and the start of Job x.

If time windows become so constrained that the latest start time for a job is before the earliest start time (LST(x) < EST(x) or alternately ΔSMAX(x,x) < 0) then we can infer that there is no feasible time at which Job x can be scheduled for which it satisfies all of its time constraints (in other words, there has been some logical contradiction implied in specifying the time constraints).

4.2.4 Algorithms

Shortest path problems can generally be solved using algorithms such as Floyd's or Dijkstra's (revised for graphs with negative cost arcs) [Larson and Odoni, 1981]. There are several special properties of this constraint problem which can be exploited to yield more efficient implementation.

Floyd's algorithm has the advantange of being very easy to program (it requires only two short lines of code in the APL computer language). Floyd's algorithm involves on the order of n cubed computation, where n is the number on nodes in a graph. It produces the shortest distance from all nodes to all other nodes. Further, there is the added advantage that as new arcs are added to a network, (i.e., by adding a new constraint), the table of shortest paths can be updated with only on the order of n squared computations. This can be shown by examining Floyd's algorithm. Let D_i be the table of shortest paths at iteration i of the algorithm, and let D_0 be the initial ΔMAX matrix produced in step 4, as described above. Then Floyd's algorithm is:

STEP 1: Set i = 1

STEP 2: Set $D_i = \text{Min}\ \{D_{i-1}, B_i\}$

where the minimization is across corresponding elements of D_{i-1} and B_i and where B_i is defined as the generalized outer product with respect to addition of the ith column of D_{i-1} and the ith row of D_{i-1}.

$$\text{i.e.,}\quad B_i(j,k) = D_{i-1}(j,i) + D_{i-1}(i,k)$$

STEP 3: If i = n stop; if i < n then set i = i + 1 and go to Step 2.

Step 2 of the algorithm involves n squared computations and it is repeated through n iterations; hence the algorithm is of the order of n cubed.

To incorporate (and thereby propagate) a new arc (constraint) $J'(x,y) = \mu$ into a reduced shortest path matrix J (i.e., one which is the result of Floyd's algorithm), where $\mu < J(x,y)$, all one has to do is run Step 2 of the algorithm twice on J': once with i=x and once with i=y.

That this procedure results in the new updated shortest path matrix can be seen from the following argument:

1) There is nothing inherently special in the numbering of the nodes in the algorithm, hence we can renumber the nodes arbitrarily, e.g., one can rename node 1 as node 6 and node 6 as node 1.

2) Applying Step 2 of Floyd's algorithm to J' for any i not equal to x or y will not change J', because Step 2 will change J' only if elements in row i or column i have been changed (which is only true for i = x or i = y). This is because J was already a shortest path matrix, and hence applying Step 2 to J for *any* i will not change J.

3) As there is nothing inherently special about the numbering of the nodes, one can renumber the nodes such that node x becomes node n-1 and node y becomes node n.

4) Applying Floyd's algorithm to this renumbered matrix will produce no changes for the first n-2 iterations; the only ones that will produce any changes are the last two iterations.

5) Reordering the rows and columns of a matrix will have no effect on Step 2 (the generalized outer product with respect to addition), except that the resulting matrix will also be similarly reordered. Hence it is not really necessary to run Floyd's algorithm incrementing i by 1 at each iteration; all that is really necessary is that on each iteration i be chosen as some integer between 1 and n without duplication. Therefore it is also not necessary to renumber the nodes; the shortest path matrix can be updated directly. The two iterations of Floyd's algorithm require n squared computations, as compared to the order of n cubed operations which would be needed to entirely repropagate all of the constraints.

4.2.5 Implementation Considerations

In implementation of the crew activity planning problem, constraints are added interactively by the user, and not in a batch. Hence, the shortest path matrix can be updated after each constraint is added, with n squared computations. For reasonable problems, this amount of computation can be done "in real time" so that a user can immediately see the effects of each new constraint on the shortest path matrix (and hence on earliest and latest start times). It is true that in the worst case, there can be n squared constraints, thus resulting in the order of n to the fourth computations, but in practice there are far fewer constraints (and the delay for each constraint addition in an interactive environment is negligible anyway).

In addition to maintaining the ΔMAX matrix, it is also necessary to maintain a record of the latest end times and the precedence constraints. This is because these constraints contain more restrictions upon the schedule than is included in their transformation to ΔMAX constraints, as described previously.

In order to maintain the latest end time constraints, a vector LET can be constructed, where LET(x) is the latest end time of Job x. This vector is initialized to LET(x) = END and then each entry is replaced by any explicitly entered value for LET(x) whenever this value is less than the current value. Inference can also be used to infer a reduced value for LET(x) according to the formula: LET(x) = MIN {LET(x), LST(x) + MAXT(x), LST(y) for all y such that P(x,y) = 1}. With interactive entering of constraints, this calculation may have to be performed after each new constraint is entered, because any new constraint can potentially impact the latest start times of some relevant job. By requiring that each job start after EST(x), start before LST(x), and end before LET(x) (as found from the ΔSMAX matrix and the LET vector) the full inference from the constraints is fully realized.

As jobs become scheduled, one can then make further inference based on the now known start and end times of the job. For example, suppose Job x becomes scheduled with start time ST and end time ET. Note that ST ≥ EST(x), ST ≤ LST(x), ET ≤ LET(x), and MINT(x) ≤ (ET - ST) ≤ MAXT(x). If (ET - ST) > MINT(x), we can then make a greater inference from P(x,y) = 1 than ΔMIN(x,y) = MINT(x). This is effectively done by changing MINT(x) (and MAXT(x)) to ET - ST and then adding and propagating new constraints ΔMIN(x,y) = ET - ST (i.e. ΔMAX(y,x) = -μ) for all y such that P(x,y) = 1. We also add and propagate new (tighter) constraints EST(x) = ST, LST(x) = ST, and LET(x) = ET. While the propagation of the constraints added as a result of scheduling Job x may cause the windows for other jobs to tighten, it cannot cause them to become infeasible: if this were not true, we could have made further inferences from the performance times and constraints on the other jobs, thus making the window for Job x smaller. However, this would imply a contradiction because we have already established that the window for Job x is already as small as can possibly be inferred from the other jobs.

Through active constraint propagation, it is thus possible to maintain values for the earliest and latest possible start times and the latest possible end time for each job. Further, when the constraints are initially processed, one can check that the constraints do not cause some diagonal element of ΔSMAX to become negative, thus implying a logical contradiction. It is can thus be guaranteed that all jobs can be sequenced within their inferred time windows, observing all time constraints. Additionally, any job scheduled within its time window cannot possible force some other job to become infeasible (with respect to its time constraints).

4.2.6 Storage Considerations

The disadvantage of Floyd's algorithm is that it requires storage of n squared numbers. For n on the order of 1000, as in the crew activity scheduling problem, this requires on the order of 1,000,000 entries. It should be noted that the only results that are really needed (to find earliest and latest start times) are the 2n entries in the first row and first column of ΔSMAX. Dijkstra's algorithm, on the other hand, has little stowage requirement beyond the explicit list of constraints and the initialized values of ΔMAX(*,x) and ΔMAX(x,*). Dijkstra's algorithm, as modified to allow for arcs of negative length, requires order of n cubed computations for finding the shortest path from any given node to all other nodes, and it must be used twice: once to find the shortest path from the source node (*) to all other nodes, and once to find the shortest path from all nodes to the source node (this is done by reversing the directions of all the arcs and then finding the shortest path from the source to all other nodes). There are several other disadvantages to Dijkstra's algorithm: it is difficult to program, and there is no analogous way to update the shortest paths as with Floyd's algorithm; adding a new arc can necessitate repeating the entire algorithm.

Floyd's algorithm can be modified to require less storage provided that the problem is decomposable into separate projects (see Section 4.3.1). The problem can be decomposed if groupings of jobs can be found for which the constraints on each of the jobs in the group pertain only to * and not to other jobs in other groups (this will often be the case, as the ΔMAX matrix will usually be sparse of significant information other than in the * row and column). Consider the case where there are 5 jobs in addition to *, and where there are constraints relating Jobs 1, 2, and 3, and other constraints relating only Jobs 4 and 5. Then all that is required is two matrices, one 4 x 4 (containing *, 1, 2, and 3) and the other 3 x 3 (containing *, 4, and 5). This requires a storage of 25 numbers as opposed to the 36 numbers required for the full 6 x 6 matrix. Savings can be much greater by decomposing larger matrices, although in the worst case all jobs will be interrelated, and the full n x n matrix is necessary.

To implement this modification, one would start with a 2 x 2 matrix for each job (containing * and the job). Constraints relating the jobs to * can then be added directly to the job's matrix. When constraints relating jobs to other jobs are processed or added, new matrices are formed by merging the two 2 x 2 matrices for the two jobs, thereby creating a 3 x 3 matrix. Henceforth, when any constraints are found relating these same jobs, the matrix can be updated directly (note that this updated computation is now on the order of the square of the size of this smaller matrix, not the order of n squared). Whenever constraints relate jobs in different matrices, the matrices are merged, as above. It should be noted that the storage requirements can be further reduced by not storing the diagonal elements in the original 2 x 2 matrices, as they are zero.

Merging the matrices is accomplished in a straightforward manner. Consider the previous example with one matrix containing Jobs *, 1, 2, and 3 and the other matrix containing Jobs *, 4, and 5. Adding a constraint relating Jobs 2 and 5 will necessitate merging the two matrices into a 6 x 6 matrix containing Jobs *, 1, 2, 3, 4, and 5. The positions for which there are no data, e.g. (2,4), are set to infinity (or some suitably large number). Then to update the matrix, all one has to do is run Step 2 of Floyd's algorithm three times: once with i=* and once each for i equal to the two jobs of the new constraint (in this case i=2 and i=5). Merging is thus also an order of n squared computation.

4.2.7 A Worked Example of Constraint Propagation

Consider the following scheduling problem:

START time of schedule = 0
END time of schedule = 20
There are 5 jobs, each requiring a single crewmember. There are 3 crewmembers.
Crewmember 1 performance times on each of the jobs are {2, 6, 3, 12, 8}
Crewmember 2 performance times on each of the jobs are {3, 5, 3, 14, 7}
Crewmember 3 performance times on each of the jobs are {4, 5, 3, 11, 8}
The minimum durations for each of these jobs are then MINT = {2, 5, 3, 11, 7}
and the maximum durations are MAXT = {4, 6, 3, 14, 8}

The following time constraints apply:

EST(1) = 2	Job 1 must start no earlier than time 2.
LST(4) = 6	Job 4 must start no later than time 6.
LET(3) = 17	Job 3 must end no later than time 17.
LET(4) = 15	Job 4 must end no later than time 15.
P(1,5) = 1	Job 5 must start after the end of Job 1.

CO(2,3) = 1	Job 2 and 3 must start concurrently.
ΔMIN(4,1) = 1	Job 1 must start at least 1 time unit after Job 4.
ΔMAX(4,5) = 10	Job 5 must start no later than 10 time units after Job 4.

Constraint reduction (Section 4.2.2) can now be performed:

STEP 1: Add an event * with MINT(*) = MAXT(*) = 0 and required start time RST(*) = 0.

STEP 2: Change all constraints to ΔMIN and ΔMAX constraints:

EST(1) = 2 becomes ΔMIN(*,1) = 2.
LST(4) = 6 becomes ΔMAX(*,4) = 6.
LET(3) = 17 becomes ΔMAX(*,3) = 17 - MINT(3) = 14.
LET(4) = 15 becomes ΔMAX(*,4) = 15 - MINT(4) = 4.
P(1,5) = 1 becomes ΔMIN(1,5) = MINT(1) = 2.
CO(2,3) = 1 becomes ΔMIN(2,3) = 0 and ΔMAX(2,3) = 0.
ΔMIN(4,1) = 1 and ΔMAX(4,5) = 10 remain unchanged.

STEP 3: Change all ΔMIN constraints to ΔMAX constraints:

ΔMIN(*,1) = 2 becomes ΔMAX(1,*) = -2
ΔMIN(1,5) = 2 becomes ΔMAX(5,1) = -2
ΔMIN(2,3) = 0 becomes ΔMAX(3,2) = 0
ΔMIN(4,1) = 1 becomes ΔMAX(1,4) = -1

STEP 4: Form the ΔMAX matrix:

Initializing ΔMAX(k,*) = 0, k = *, 1, 2, 3, 4, 5
ΔMAX(k,k) = 0, k = 1, 2, 3, 4, 5 {no job can start before time 0}
ΔMAX(m,n) = 20 - MINT(n), m = *, 1, 2, 3, 4, 5, n = 1, 2, 3, 4, 5, m≠n.
{all jobs must end before time 20}

So

$$\Delta MAX_{initial} = \begin{matrix} 0 & 18 & 15 & 17 & 9 & 13 \\ 0 & 0 & 15 & 17 & 9 & 13 \\ 0 & 18 & 0 & 17 & 9 & 13 \\ 0 & 18 & 15 & 0 & 9 & 13 \\ 0 & 18 & 15 & 17 & 0 & 13 \\ 0 & 18 & 15 & 17 & 9 & 0 \end{matrix}$$

(Note that the row (or column) corresponding to, say, Job 4, is the 5th row (or column), because Job * occupies the 1st row and column.)

Each of the ΔMAX values (5 from Step 2 and 4 from Step 3) are then added to this matrix whenever they are less than the corresponding value already entered in the matrix. So

$$D_0 = \Delta MAX = \begin{matrix} 0 & 18 & 15 & 14 & 4 & 13 \\ -2 & 0 & 15 & 17 & -1 & 13 \\ 0 & 18 & 0 & 0 & 9 & 13 \\ 0 & 18 & 0 & 0 & 9 & 13 \\ 0 & 18 & 15 & 17 & 0 & 10 \\ 0 & -2 & 15 & 17 & 9 & 0 \end{matrix}$$

Figure 4.2 graphically shows all the arcs which are from explicitly specified time constraints (labelled with an outline font), and arcs connected to * which result from the fact that all jobs must start no earlier than time zero nor end later than time 20 (labelled with a normal font).

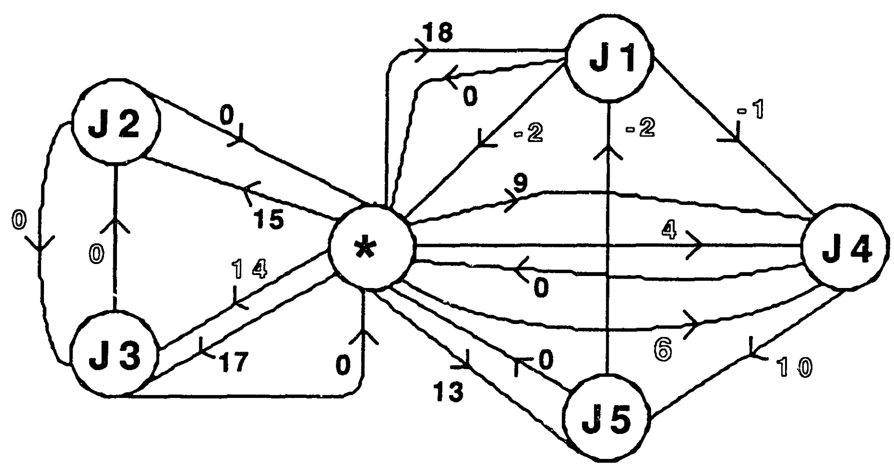


Figure 4.2: Time Constraints Configured on a Shortest Path Network

Floyd's Algorithm (as described in Section 4.2.4) can now be applied to this graph. D_0 is the ΔMAX matrix shown above.

STEP 1: Set i = 1.

STEP 2: Compute B_1. $B_1(j,k) = D_0(j,1) + D_0(1,k)$

$$B_1 = \begin{matrix} 0 & 18 & 15 & 14 & 4 & 13 \\ -2 & 16 & 13 & 12 & 2 & 11 \\ 0 & 18 & 15 & 14 & 4 & 13 \\ 0 & 18 & 15 & 14 & 4 & 13 \\ 0 & 18 & 15 & 14 & 4 & 13 \\ 0 & 18 & 15 & 14 & 4 & 13 \end{matrix}$$

Compute $D_1 = \text{Min}\ \{D_0, B_1\}$

$$D_1 = \begin{matrix} 0 & 18 & 15 & 14 & 4 & 13 \\ -2 & 0 & 13 & 12 & -1 & 11 \\ 0 & 18 & 0 & 0 & 4 & 13 \\ 0 & 18 & 0 & 0 & 4 & 13 \\ 0 & 18 & 15 & 14 & 0 & 10 \\ 0 & -2 & 15 & 14 & 4 & 0 \end{matrix}$$

STEP 3: As i < 6, increase i by 1 (so i = 2). Return to Step 2.

The algorithm will continue until i = 6. At this point ΔSMAX, the shortest path matrix (containing the shortest path from each node to each of the other nodes in Figure 4.2), will be computed:

$$D_6 = \Delta SMAX \quad \begin{matrix} 0 & 11 & 14 & 14 & 4 & 13 \\ -2 & 0 & 12 & 12 & -1 & 9 \\ 0 & 11 & 0 & 0 & 4 & 13 \\ 0 & 11 & 0 & 0 & 4 & 13 \\ 0 & 8 & 14 & 14 & 0 & 10 \\ -4 & -2 & 10 & 10 & -3 & 0 \end{matrix}$$

This matrix does not contain any negative entries along the main diagonal, and there are therefore no logical contradictions implied in the constraints. It is thus assured that all the jobs can be sequenced to meet all their time constraints within the 20 time unit planning window (without consideration of the limited number of crewmembers or of any limited resources).

The first row of this matrix contains the inferred (from constraint propagation) latest start times of each of the jobs, and the first column contains the negatives of the inferred earliest start times.

The latest end times for each Job x can also be computed by taking the minimum of: 1) the END time of the schedule (in this case, LET(x) ≤ 20, x = 1, 2, 3, 4, 5); 2) any explicitly specified latest end time for Job x (in this case, LET(3) ≤17 and LET(4) ≤ 15); 3) the inferred latest start time of Job x, plus the maximum duration of Job x (in this case, LET(x) ≤ LST(x) + MAXT(x), x = 1, 2, 3, 4, 5); and 4) the inferred latest start times of any jobs which must follow Job x because of precedence constraints (in this case, LET(1) ≤ LST(5)). It can be seen that in order to find the latest end times, it is necessary to retain the original information regarding the latest end time constraints and the precedence constraints, because, as stated in Section 4.2.2, the transformations of these constraints into ΔMIN constraints specify necessary but not sufficient conditions to completely describe these constraints. Therefore, for this problem, one obtains:

EST (1) = 2	LST(1) = 11	LET(1) = 13
EST(2) = 0	LST(2) = 14	LET(2) = 20
EST(3) = 0	LST(3) = 14	LET(3) = 17
EST(4) = 0	LST(4) = 4	LET(4) = 15
EST(5) = 4	LST(5) = 13	LET(5) = 20

Suppose now that it is decided to have Job 1 performed by Crewmember 2, who has a performance time of 3. It is also decided to have this Job start at time 4, and thus end at time 7. As described in Section 4.2.5, active constraint propagation can now be utilized to make further inference based upon this information. This is done by adding the new constraints EST(1) = 4, LST(1) = 4, LET(1) = 7, and ΔMIN(1,5) = 3 (from the precedence constraint). The minimum and maximum completion times for this job are also updated so that MINT(1) = MAXT(1) = 3.

These new constraints can now be converted to ΔMAX constraints:

ΔMAX(1,*) = -4

ΔMAX(*,1) = 4

ΔMAX(5,1) = -3

Inserting these values into the old ΔSMAX matrix:

$$\Delta MAX = \begin{matrix} 0 & 4 & 14 & 14 & 4 & 13 \\ -4 & 0 & 12 & 12 & -1 & 9 \\ 0 & 11 & 0 & 0 & 4 & 13 \\ 0 & 11 & 0 & 0 & 4 & 13 \\ 0 & 8 & 14 & 14 & 0 & 10 \\ -4 & -3 & 10 & 10 & -3 & 0 \end{matrix}$$

The effects of these new constraints can now be propagated by using Step 2 of Floyd's shortest path algorithm 3 times: once with i = 1 (for event *); once for i = 2 (for Job 1); and once for i = 6 (for Job 5). This give the new ΔSMAX matrix:

$$\Delta SMAX = \begin{matrix} 0 & 4 & 14 & 14 & 3 & 13 \\ -4 & 0 & 10 & 10 & -1 & 9 \\ 0 & 4 & 0 & 0 & 3 & 13 \\ 0 & 4 & 0 & 0 & 3 & 13 \\ 0 & 4 & 14 & 14 & 0 & 10 \\ -7 & -3 & 7 & 7 & -4 & 0 \end{matrix}$$

As before, earliest start times, latest start times, and latest end times can now be computed. Note that in addition to the changes in the data for Job 1, the earliest start time of Job 5 and the latest start time of Job 4 have also been affected.

EST (1) = 4	LST(1) = 4	LET(1) = 7
EST(2) = 0	LST(2) = 14	LET(2) = 20
EST(3) = 0	LST(3) = 14	LET(3) = 17
EST(4) = 0	LST(4) = 3	LET(4) = 15
EST(5) = 7	LST(5) = 13	LET(5) = 20

4.3 Heuristic Dispatching Techniques

Extensive literature has been compiled detailing, analyzing, and comparing algorithms for dealing with scheduling problems. As the crew activity planning problem is in the class NP-complete, all optimal algorithms will probably be exponential (see Section 4.1). Exact techniques [Patterson, 1984] utilizing branch-and-bound [Stinson, et al., 1978], dynamic programming, linear programming, integer programming [Talbot and Patterson, 1978], quadratic programming, and zero-one programming [Pritsker, et al., 1969], as well as bounded enumeration [Davis, 1969; Davis and Heidorn, 1971] have been used to obtain some improvement over explicit enumeration techniques, but this problem still is intractable for all but the smallest formulations. For example, the zero-one programming formulation that has been proposed for the crew activity planning problem would require the use millions of variables [Mathis, 1981]. It is therefore necessary to turn to heuristic techniques for finding "good" (i.e. near-optimal) solutions.

Virtually all the heuristics developed for solving scheduling problems (as well as the ones developed in this thesis) have been dispatching algorithms. In a dispatching algorithm, some job is selected for scheduling, and then added to a partial schedule. The next job is then selected for scheduling, until all jobs have been scheduled. At each stage the priority of each job is determined, and the job with the largest priority is chosen. Serial dispatching techniques predetermine the ordering of all jobs before any scheduling begins. In parallel dispatching techniques, the next job is not selected until the previous job has been scheduled. Parallel dispatching algorithms operate on more information (i.e., the partial schedules) than serial dispatching algorithms, and superior results can usually be achieved. It should be noted that the priority of a job, as determined by a heuristic, is not the same as the weight of the job, w_j (as defined in Section 3.3.1), although a job's weight may be a factor in determining its priority.

In addition to a heuristic for determining which job is to be scheduled next, multi-processor scheduling problems also require a heuristic for deciding which processor (crewmember) to assign to a job, as well as the time in the schedule at which the job is to be assigned. It should be noted that even if all potential job orderings are tried, an optimal solution might not be found unless all possible combinations of crew selection are also investigated for each ordering.

4.3.1 Previous Research: Resource Constrained Multi-Project Scheduling

Unfortuately, most of the research investigated in the literature addresses what is known as resource constrained multi-project scheduling, a problem much simpler than the crew activity planning problem. Hundreds of papers have been written on the resource constrained multi-project scheduling problem, and several studies have detailed and compared various heuristic and optimal algorithms that have been developed to solve this problem [Davies, 1973; Davis, 1973a; Patterson, 1973; Davis and Patterson, 1975; Patterson, 1976; D. Cooper, 1976; Kurtulus and Davis, 1982].

In multi-project scheduling, each project consists of a set of jobs which are entirely linked together by precedence constraints. The structure imposed by these precedence constraints greatly simplifies the scheduling problem (at least, compared to the crew activity scheduling problem described above). The only other types of constraints usually allowed in the resource constrained multi-project scheduling model (in addition to precedence constraints) are renewable resource constraints, although some models can accommodate earliest start time constraints and latest end time constraints. Additionally, resource usage by each job is always a constant, and resource limits are usually not time varying either. The model usually does not include any of the other types of time or target constraints defined in Section 3.3.1, or the nonlinear or non-renewable types of resource usage previously mentioned [exceptions are Slowinski, 1980; Altman, et al., 1970]. The use of heuristic algorithms from the literature that have not been designed to analyze many of the constraints present in the crew activity planning problem could prohibit efficient usage of resources and the formation of optimal schedules, or could even lead to the production of infeasible schedules.

This is because in the resource constrained multi-project scheduling problem, the heuristics used have no trouble finding a complete schedule. The goal is then to find a complete schedule which is optimal. Many of the time constraints present in the crew activity planning problem are absent. These constraints, such as latest start time constraints, latest end time constraints, concurrence constraints, minimum and maximum time intervals between jobs, nonrenewable resource constraints, target constraints, and the limited numbers of crewmembers, can easily cause a partial schedule to become infeasible with respect to a specific job. Additionally, in the crew activity planning problem, there are often more jobs included in the problem statement than could possibly be scheduled, so the goal is to find a feasible (but incomplete) schedule which, for example, maximizes the weights of the jobs successfully scheduled. Most of the heuristic rules in the literature incorporate no methodology for dealing with situations where some jobs can not be included in a schedule and where tradeoffs between jobs are necessary.

Another difference between the two scheduling problems is largely conceptual, but has important repercussions on the types of heuristic algorithms which will solve the problems effectively. In resource constrained multi-project scheduling problems, the ratio of the number of jobs to the number of projects is usually quite large; typically greater than ten. In the crew activity planning problem, we can extend the idea of a project to include a subset of the jobs which are related to each other by any type of time constraint which relates one job to another. In addition to precedence constraints, this would include concurrence constraints, and constraints relating minimum and maximum differences in start times between the jobs. In spite of this more generalized definition of a project, the general crew activity planning problem can involve consideration of problems where the ratio of jobs to projects is quite small, even one.

An additional difference between resource constrained multi-project scheduling and crew activity planning is that, in the multi-project scheduling model, all jobs require exactly one processor, all processors can complete any given job in the same amount of time, and the number of processors available is usually assumed to be unlimited. (Actually, some versions of the resource constrained multi-project scheduling problem can accommodate a limited number of processors, by considering the processors to be a limited resource. In the crew activity planning problem, however, it is not possible to consider crewmember usage as a resource, at least in the traditional sense, because crewmembers may have differing processing times for the same job or may even be unrated for a particular job. Hence, knowing that there is "one unit of crewmember time" available during an interval does not specify important information such as which crewmember is available or even if the interval is continuous with the same crewmember). All of the heuristics for solving multi-project scheduling do not incorporate any methodology for considering the tradeoffs which must be made when a job can be completed by different crewmembers in different amounts of time. Further, the crew activity planning problem may require that some jobs require more than one crewmember or that specific crewmembers be prohibited from performing a certain job. No method for dealing with these situations is incorporated into the heuristics in the literature.

A final difficulty faced by most of the heuristic techniques that have been proposed is that they are somewhat inflexible, relying on a fixed set of rules to produce a single candidate schedule. In good schedules, the jobs will often fit together like pieces of a jigsaw puzzle. While a heuristic may produce a candidate schedule of "good" quality (compared to, say, schedules produced by some random sequencing technique), the heuristic may miss the fact that a small change in the schedule might improve results significantly. A heuristic methodology which could effectively make changes in schedules in order to seek improvements would be of great benefit to both multi-project scheduling and crew activity planning.

4.3.2 Previous Research: Crew Activity Planning

Perhaps the most applicable research pertaining to space station crew activity planning are three studies commissioned by the NASA Marshall Space Flight Center during 1980, 1981, and 1986 [Grone and Mathis, 1980; Mathis, 1981; Deuermeyer, et al., 1986]. It should be noted that while the models addressed by these studies were much more robust than the resource constrained multi-project scheduling problem, they do not encompass the entire crew activity planning model outlined in Section 3.3.1: until now, this problem has been considered too complex to solve in full. As a result, artificial constraints and approximations were sometimes added to "force" a real-world problem to fit the model. This practice of adding non-real constraints could cause the unintentional elimination of good solutions. In addition, considerable hand editing of schedules was necessary to in order make sure that there was no violation of the constraints which could not be incorporated into the model. Grone and Mathis describe the dispatching system used by NASA and the problem which they were addressing:

> "The problem under consideration is that of scheduling a set of experiments to be performed on a given Spacelab mission. Through a preliminary analysis, which is not discussed here, a set of compatible experiments is compiled which are hopefully to be included in a specific mission. The experiments are then converted into model sheets by the principal investigator, who is responsible for the experiment design, and a NASA project engineer familiar with the format and requirements of the model sheets. The model sheet divides each experiment into a sequence of steps, each of which has certain requirements in terms of crew, equipment, and energy usage; and may in addition require that the Spacelab be in a certain position or configuration, or have certain targets available. Examples of targets are planets, communication satellites, and data receiving facilities on the earth's surface. If the experiment is to be performed more than once the number of performances desired is included on the model sheet. A typical Spacelab mission lasts for seven days and involves six crewmembers.
>
> "Extremely sophisticated software is available to assist in scheduling of the experiments. The current program is designated by TLP for timeline program. This program takes the model sheets in some order and "front loads" them in this order. By <u>front load</u>, it is meant that the model is scheduled at the earliest possible time which satisfies all the model constraints and which does not conflict with the requirements of previously scheduled models. If it is impossible to schedule a certain model at any time the program then goes to the next model in the sequence.

This program is currently being upgraded to a program denoted by ESP for experiment scheduling program, which has the capability of either front loading or back loading any given model, and which is to supersede TLP sometime in 1981.

"Even with the extremely advanced and effective software available, the problem of scheduling the experiments in an optimal way currently requires an enormous amount of pre-flight man-hours. By an optimal schedule, we roughly mean one that includes the maximal number of experiments and performances, and which makes maximal use of the available resources. Since each set of models designated for inclusion in a specific mission are only roughly sorted for compatibility, it is frequently impossible to include every performance of every experiment.

"The method previously employed has been to feed TLP various random orderings of the model sheets and have the project engineers examine the characteristics of the resulting schedules. From these examinations other orderings are suggested and fed to TLP. After months of such trial and error, the best resulting schedule is then modified and edited by hand by a group of NASA personnel familiar with the characteristics of the given mission. The current schedule for Spacelab mission one has been under development for five years. It is clear that such a time expenditure is not only economically prohibitive, but is unacceptable in light of the fact that the Spacelab program eventually envisions flying several missions a year." [Grone and Mathis, 1980]

Mathis and Grone employed an intricate multi-attribute ranking algorithm in the 1980 study and achieved good success. While their algorithm produced somewhat poorer results for Spacelab mission 1 than the best that had been previously generated (after several years of effort), they were able to produce schedules superior to NASA's best existing schedules for Spacelab missions 2 and 3 [Grone and Mathis, 1980]. Mathis then tested an algorithm employing zero-one programming techniques, which, while much slower than TLP, produced results only comparable to TLP, and significantly poorer than the ranking algorithm [Mathis, 1981].

In the 1986 study at Texas A&M University, Deuermeyer, Shannon, and Underbrink sought further improvements upon the ranking algorithms of Mathis and Grone. Focusing upon methods for establishing a selection list that would work in harmony with ESP, they found that they could divide experiments into three categories:

Experiments in the first category consist of those experiments which must precede other experiments (by virtue of precedence constraints). Also included are experiments required to be concurrent with these experiments. There exists sophisticated and automated scheduling procedures (taken from the resource constrained multi-project scheduling literature) for solving these problems;

Experiments in the second category consist of experiments which are required to be concurrent with each other. The study found that these experiments could be sequenced randomly with no significant impact on schedule quality;

Experiments in the third category (all other experiments) were found to be best scheduled based on clustering and ranking procedures, utilizing the engineering judgement of the scheduler.

The study found that good results could be obtained by separately sequencing the experiments in each of these three categories, and then first scheduling the experiments in the first category, followed by those in the second category, and finally the experiments in the third category.

It should be noted that the Soviet Union also employs a dispatching technique for the scheduling of their Salyut work schedules. The parameters relating to each of the activities are generated and each of the experiments is given a priority. Then experiments are then spread through the flight time, starting with the experiment that has the highest priority [Blagov, 1983]. Because the list of experiments proposed is usually too long to be fully accommodated, experiments which do not fall within the flight plan are listed as reserve experiments.

4.4 Solution Techniques

Four heuristic approaches were developed to attempt to cope with the problems discussed in Section 4.3.1. Only the first two of them have previously been discussed extensively in the literature. Central to all of these approaches is the notion that no one fixed scheduling rule will always produce good schedules; some attempt must be made to (intelligently) search the solution space (i.e., the configuration space) for good solutions. Unlike the problem of minimizing a continuous function, the configuration space of combinatorial minimization problems is discrete, not a simple continuous N-dimensional space. The number of elements in the configuration space is factorially large, so that they cannot be explored exhaustively, except for the smallest problems. As the space is discrete, it is difficult to try to "continue downhill in a favorable direction." Indeed, the concept of "direction" can be difficult or impossible to define for a combinatorial minimization problem [Press, et al., 1986].

Chapter 5 discusses the implementation details of those methodologies which were judged promising, and presents emperical results validating those models.

4.4.1 Use of Multiple Heuristics

Instead of employing a single heuristic to find a solution, a viable approach is to employ many different heuristics and then take the best solution produced by any of them [Davis and Patterson, 1975; Patterson, 1976; D. Cooper, 1976]. Patterson compared the performance of many different heuristics and then performed a regression analysis to correlate the efficiency of various heuristics with many different problem parameters. In an interesting result, he found that using the rule which was in general found best for solving multiproject problems would, on average, result in an increase of 16% (in total project delays) over the best schedules produced by trying all of the eight heuristics tested. If, instead, each problem is solved using the heuristic projected best for that problem, then results obtained are, on average, only 8% worse than what could be obtained by using all heuristic procedures. From a practical point of view, however, the cost of programming and running several different heuristics is quite small compared to the costs of data acquisition, preparation, and the potential savings involved in finding superior solutions.

4.4.2 The Sampling Method

Another approach is known as the sampling method: instead of determining fixed priorities for each job, the heuristic rule is used to determine the probability that each job will be scheduled next [see, for example, D. Cooper, 1976]. This allows an intelligently directed, semi-random, search for a good solution. For example, if three jobs were to be scheduled, and a heuristic determined that these three jobs had relative priorities of 20, 50, and 30, then a standard serial dispatching techniques would schedule the jobs in the order 2, 3, 1. On the other hand, the sampling method would assign a 20% probability that Job 1 would be the first job chosen, a 50% probability that Job 2 would be the first job chosen, and a 30% probability that Job 3 would be the first job chosen. One would then repeatedly find schedules for the same problem, keeping the best. Cooper examined the resource constrained project scheduling model, and compared the schedules found (using many different heuristic rules) for both the standard parallel dispatching method and the best of 100 runs of the sampling method utilizing the same heuristic rule. This comparison was made over eight different projects and many different heuristic rules. An average improvement of 7% was found in the quality of the best schedules produced by the sampling method.

It should be noted that if the heuristic being used assigns the same priority to all jobs, then this method becomes a true random search.

4.4.3 Searching the Weighting Space

Some researchers have tried to determine job priorities by weighting together a large number of factors pertinent to each job [see, for example, Mathis and Grone, 1980]. Mathis and Grone suggested (but did not implement) a scheme whereby the weighting factors could be determined by examining the specific characteristic of a particular problem. Although this is still basically a "single shot approach," it could be extended by systematically varying the relative weights of each of the factors and then making several attempts at forming a schedule, searching through a search space defined by the weighting factors.

4.4.4 Making Perturbations in Previous Schedules

In this method, some initial schedule is determined by some heuristic or even by a random dispatching of the jobs. Once this initial schedule is formed, successive scheduling attempts are made which focus on perturbations in the "neighborhood" of this initial schedule. Techniques of this type are examples of "hill climbing," where one attempts to look for better solutions near previous solutions. For example, given the dispatcher ordering of the jobs in an initial schedule, one might try looking at schedules in which this ordering is changed by switching successive pairs of jobs. This would procede until some better schedule is found (thus restarting the algorithm) or until all pairs of jobs have been switched without improvement. Such a schedule would be said to be 2-optimal with respect to the dispatcher ordering.

Another example of this methodology might be to establish priorities for each of the jobs and then, on each iteration, increase the priorities of the jobs which become infeasible with respect to a candidate schedule. Then on the next scheduling attempt, these jobs would be scheduled earlier, and would therefore be more likely to be feasibly scheduled. In this manner, a notion of direction is established. Each new scheduling attempt moves towards a point in the configuration space (if such a point exists) for which infeasible jobs are now feasible. Of course, this could in turn cause other jobs which were previously feasible to become infeasible, and the schedule at this new point might possibly be inferior to the schedule at the old point. By repeatedly moving in directions which make infeasible jobs feasible (i.e., by increasing the priorities of those infeasible jobs), and by varying the magnitude of the move at each iteration (i.e., the amount which the job priorities are increased) it should thus be possible to search the solution space intelligently.

This methodology can be used not only to find feasible schedules, but also to drive the quality of feasible schedules towards optimality. For example, suppose the objective function is to minimize the completion time of the last job scheduled. Given a feasible schedule, one could then add a constraint to the problem requiring that all jobs end earlier than the end time of what is currently the last job scheduled. This will thus cause this last job to become infeasible, and the iterative algorithm can then be used again to find a superior schedule.

In some ways, this type of iterative algorithm is similar to what is known as the method of simulated annealing [Press, et al., 1986]. Simulated annealing, however, differs in two important characteristics. First of all, in simulated annealing no attempt is made to find intelligent directions to search. Instead, search proceeds in a random direction. The second difference in simulated annealing is that at each iteration, one does not always move to the new point in the configuration space if it does not provide a superior solution. Instead, one may move to an inferior point in the solution space with a probability which decreases as the poorness of the new solution increases.

The attractive feature of this class of algorithms is that all that is necessary in order for them to be usable is that the jobs be schedulable by a dispatcher, and that a systematic method be developed for making perturbations on the ordering from iteration to iteration. This algorithm is not dependent upon the problem structure (i.e., the types of the constraints involved) except possibly in the use of heuristics for determining an initial starting ordering for the algorithm. The difficulty in this type of algorithm is that it must be shown (empirically) that it will converge to a good solution in a reasonable amount of time. In Section 5 it is shown that variants of this algorithm converge to good solutions of the crew activity planning problem, even with random initial orderings.

Section 5: Experimental Discussion of Solution Techniques

5.1 Selection of Heuristics

5.1.1 Heuristics for Job Ordering

Ten basic heuristics were chosen to demonstrate the techniques of Section 4. The first eight of these are implementations or variations of heuristics found in the scheduling literature. These parallel heuristics are designed to choose the ordering in which jobs are dispatched. The heuristics produce a rating for each of the jobs, and then the job is chosen for scheduling which has maximum (or minimum, depending on the heuristic) rating. In each case, the heuristics are applied only among those jobs which are pending. A job is considered to be pending if there are no unscheduled jobs which must start earlier than it (i.e., job x is pending if $EST(x) \leq LST(y)$ for all unscheduled jobs y). Also, in all cases, any unresolved ties (jobs which an heuristic selects with equal rating) are broken by selecting the job with the smallest job number (i.e., Job number 4 will be selected before Job number 5, all other factors being equal).

The heuristics presented below are described assuming that each job has equal priority. In practice, the heuristics can be generalized to include a priority factor. For a heuristic which attempts to find a job with some maximum value, such as Heuristic 2, the rating for each job is simply multiplied by the job's priority, and the job with the largest resulting value is then selected. For heuristics which attempt to find a job with some minimum value, such as Heuristic 1, the rating for each job is divided by the job's priority, and the job with the smallest resulting value is selected.

To aid in the presentation of results, each of the heuristics below is given an abbreviation. The definitions of the heuristics used in this study do not necessarily exactly coincide with the definitions of similarly named heuristics in other studies.

Heuristic 1: Shortest Job First (SJF)

This heuristic is implemented by finding the pending job with the minimum duration. The duration is defined as the shortest possible completion time of the job. For example, if there are five crewmembers which can perform a job with completion times of 20, 30, 30, 40, and 15 minutes respectively, and the job requires 2 crewmembers to complete it, then the shortest possible completion time is 20 minutes (the second smallest completion time).

Heuristic 2: Longest Job First (LJF)

This heuristic is implemented by finding the pending job with the largest duration, as defined in Heuristic 1 (i.e., based on the shortest completion time by the crewmembers).

Heuristic 3: Minimum Slack Method (MSLK)

This heuristic is implemented by finding the pending job with the minimum slack, where slack is defined as the job's latest start time minus the job's earliest start time. Latest and earliest start time are dynamically (during the heuristic) determined by critical path methods as described in Section 4.2. Therefore, when time constraints are present, the slack on each job may change as other jobs are scheduled.

Heuristic 4: Greatest Resource Demand (GRD)

This heuristic selects the job with the greatest resource usage first. The relative importance of Resource z is defined as $I_z = \sum_j d_j \times r_{zj}/R_z$ where d_j is the duration of Job j (as defined in Heuristic 1), r_{zj} is the rate of usage of Resource z by Job j and R_z is the total amount of Resource z available at any time. A similar quantity can also be established for crew usage $C = \sum_j d_j \times c_j/m$ where c_j is the number of crewmembers required for each job and m is the total number of crewmembers available. Then the total resource usage of Job j, T_j is defined as:

$$T_j = d_j \left(C \times c_j/m + \sum_z I_z \times r_{zj}/R_z\right)$$

Heuristic 5: Greatest Remaining Resource Requirement (GRR)

This heuristic selects the job with the largest remaining resource requirement by adding together the total resource usages (as determined with Heuristic 4) for the job and all jobs which are constrained to start after it.

Heuristic 6: All Jobs Equally Rated (**JER**)

This heuristic rates each of the jobs equally, with the result that in the absence of prioritization, the pending job with the lowest job number is selected, utilizing the rule for breaking ties among equally rated jobs. When jobs are prioritized, the job with the maximum priority is chosen.

Heuristic 7: Minimum Latest Finishing Time (**LFT**)

This heuristic selects the pending job which has the soonest latest end time, as dynamically determined from critical path methods (see Section 4.2).

Heuristic 8: Minimum Earliest Start Time (**EST**)

This heuristic selects the pending job which has the soonest earliest start time, as dynamically determined from critical path methods (see Section 4.2).

Heuristic 9: Maximum Compatibility Method (**MCM**)

This heuristic employs a compatibility matrix which indicates the desirability of each job following each other job in the schedule (the generation of this matrix, as well as the motivation for this algorithm, is fully discussed in Section 5.1.3). For example, the entry in row 3, column 9, of this matrix indicates the compatibility of having Job 9 follow Job 3 in the schedule. If no jobs have yet been scheduled, the job is selected which has the largest entry in its row of the compatibility matrix. Subsequent jobs are chosen for scheduling by selecting the unscheduled pending job with the largest entry in the row of the compatibility matrix corresponding to the job which is completed last in the current partial schedule.

Heuristic 10: Constrained Maximum Compatibility Method (**CCM**)

This heuristic is a variation of Heuristic 9, where each column of the compatibility matrix is divided by the slack (as defined in Heuristic 3) of each job. In the event that there are pending unscheduled jobs with zero slack, then the largest entry is selected among the columns in the compatibility matrix corresponding to these jobs, as in Heuristic 9.

Figure 5.1 illustrates the way in which the heuristics for ordering the jobs are used to form a schedule. Implicit to Figure 5.1 is the methodology by which a job is added to a schedule, which is discussed in the next section.

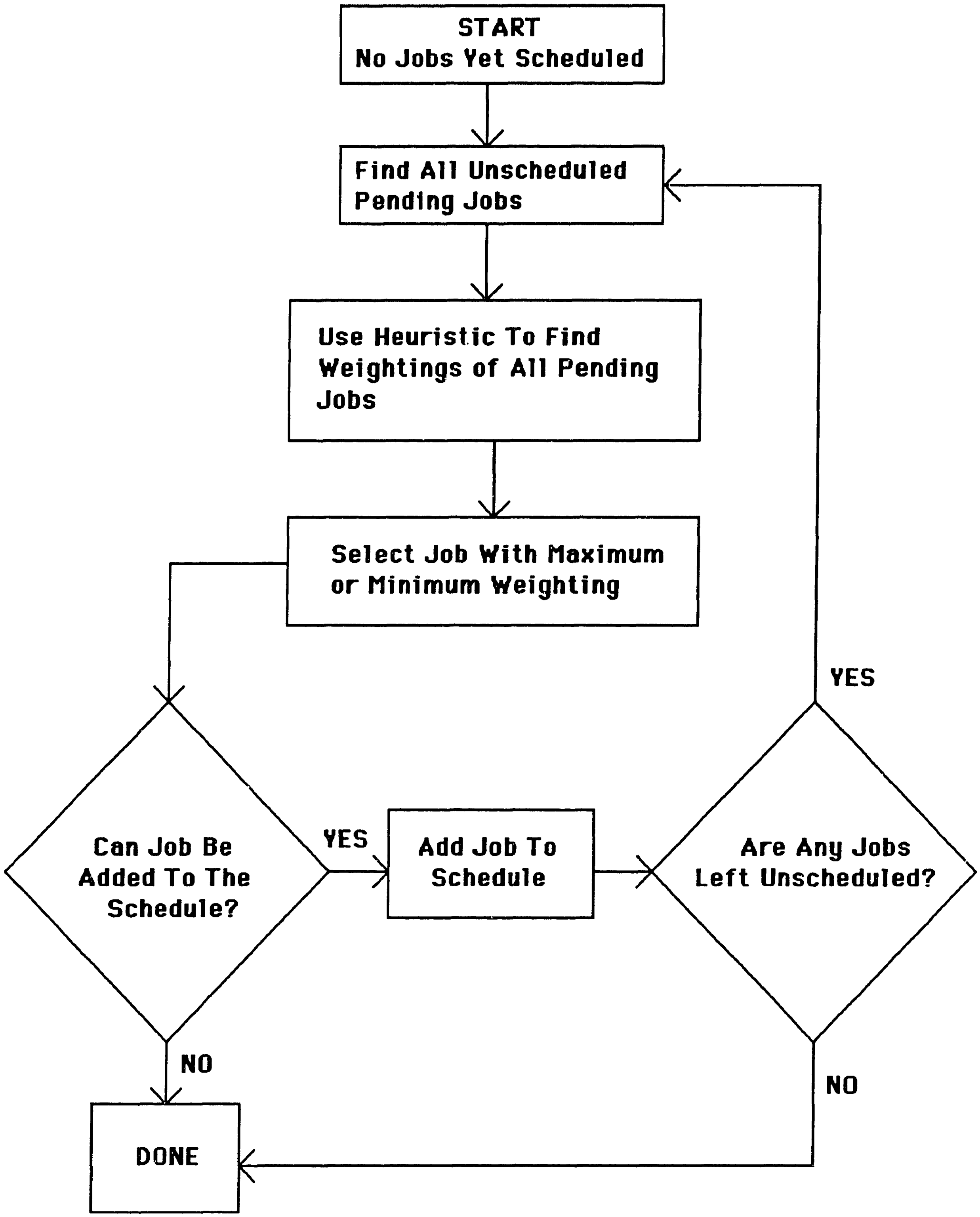


Figure 5.1: Using an Heuristic to Build a Schedule

5.1.2 Heuristics for Crewmember and Start Time Assignment

Just as heuristics are used to determine the dispatch order of the jobs, heuristics are also necessary in order to decide a job's start time and the crewmember(s) who are assigned to perform the job (note: some jobs do not require crewmembers to perform them). Several heuristics are described below for performing this function. Each of these heuristics uses some method for choosing which crewmembers to assign a job to, and then assigns the job to these crewmembers at their earliest available start time. This will often result in the filling of "holes" in the schedule; however, these heuristics make no attempt to "widen" a hole (by rescheduling previously scheduled jobs) to accommodate scheduling a job which almost fits into a gap in the schedule. The iterative algorithm discussed in Section 5.2.4.1 indirectly accomplishes this function.

Heuristic A: Minimizing Crew Workload

This heuristic assigns a job to the feasible set of crewmembers who are able to jointly complete the job and will have the minimum total workload if assigned the job. Any ties are broken by picking the crewmember with the lowest crew number. The job is started at the earliest possible time at which the crewmembers can jointly complete it.

Heuristic B: Equalizing Crew Workload

This heuristic attempts to pre-plan the schedule by making preliminary assignments of crewmembers to jobs before any jobs are yet scheduled. This pre-planning is done using some heuristic which tries to equalize individual workload while keeping total workload as minimal as possible. As the schedule is generated, an attempt is made to assign each job, at the earliest time possible, to the crewmember(s) who are indicated by the pre-planning [this heuristic was suggested by A.H.G. Rinnooy Kan, personal communication, 1984]. If during the scheduling process it should occur that it is impossible to assign the pre-selected crewmember(s) to some job, then Heuristic A is used to assign the crewmember(s) to this job, and a new pre-plan is generated for the remaining unscheduled jobs.

Heuristic C: Schedule At Earliest Possible Completion Time

This heuristic assigns each job to the crewmember(s) who can complete it earliest, at their earliest possible start time. Heuristic A is used to break any ties.

Execution of each of these three heuristics requires significantly different amounts of computation. Consider a job requiring m crewmembers out of a total of M available crewmembers. Clearly, there are $D = M!/(M-m)!m!$ combinations of crewmembers. D is thus an exponential function of the number of crewmembers, but is limited to relatively small values if the total number of crewmembers is limited to small values. In the worst case, each heuristic might require checking all D alternatives in an attempt to find a feasible schedule. Heuristics A and B only require checking alternatives until a feasible one is found. Heuristic C, on the other hand, requires checking all the D alternatives to find the one(s) which can complete it earliest.

When Heuristic A selects the crewmembers, the current (at this point in the heuristic) workload of each crewmember is added to the time required by each crewmember to complete the job, and the m crewmembers are then chosen which have the smallest resulting values. If there is no feasible scheduling time for these crewmembers, then the subset of m crewmembers with the next smallest total value (determined by adding together the values for each of the crewmembers in the subset) are tested for scheduling. The sorting of the total values can be done in ($D \times \ln D$) time if it is done all at once; if it is done on an as needed basis (i.e., finding the next best crewmember combination only if the last one was infeasible) then an order of D^2 operations are necessary to sort the D alternatives.

The effort required to use Heuristic C is essentially the same as Heuristic A, except that each of the D alternatives must always be searched. In practice Heuristic C usually takes several times longer than Heuristic A. It should be noted that this factor represents only the choosing of the crewmembers, not the entire process of selecting a job, choosing the crewmembers, and updating the schedule. Choosing the crewmembers represents a small (although still significant) part of this process (see Section 5.3.1). The variation in time required for using Heuristic C over Heuristic A does not have a big impact in total scheduling time, and is thus not an important factor in deciding which heuristic to use.

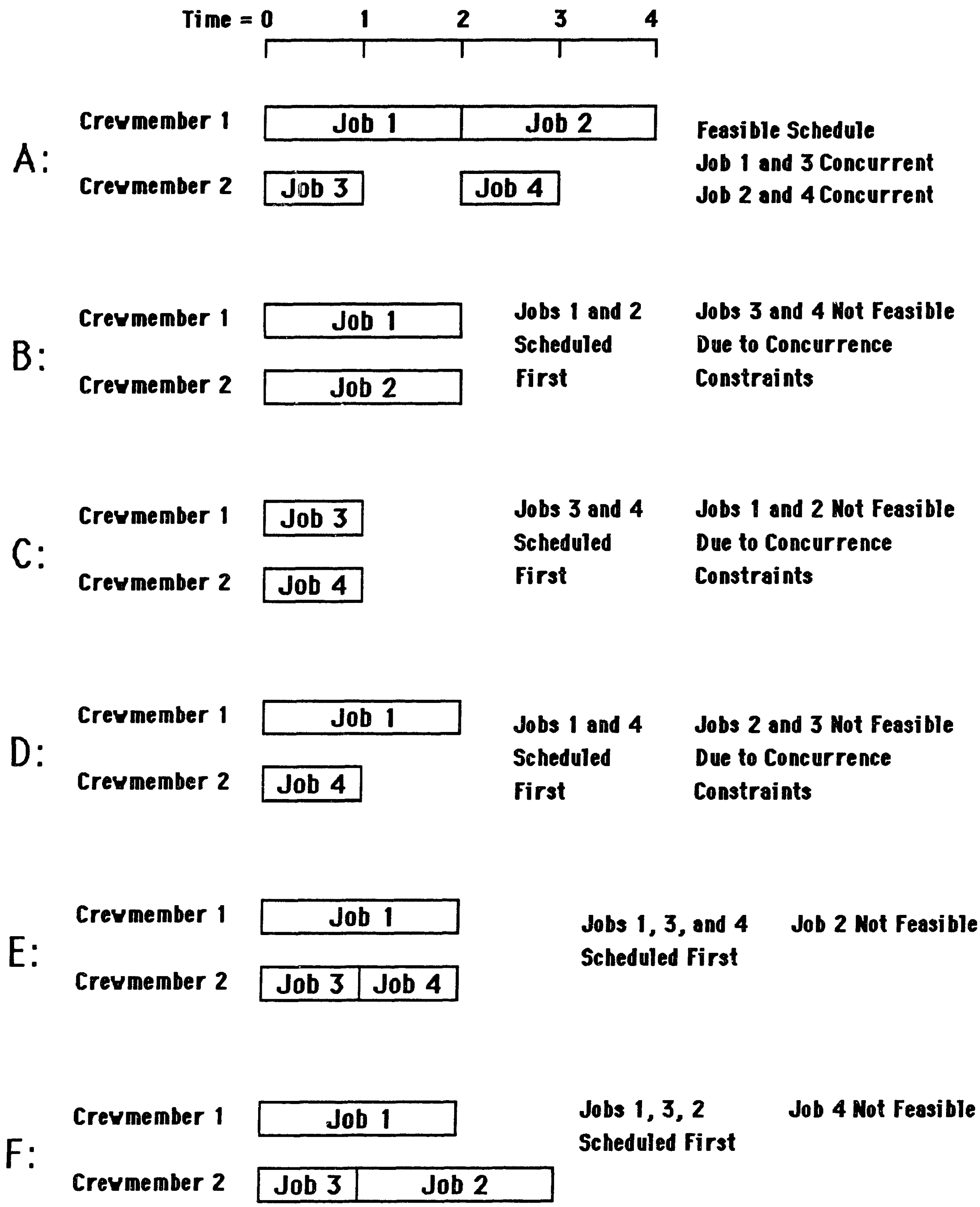
Heuristic B, on the other hand, can significantly lengthen the scheduling process. This is because pre-planning must be done in order to decide the initial (tentative) assignment of crewmembers to jobs. No matter what method is used for performing the pre-planning, this process is at least of the order of the number of jobs. Further, this pre-planning must be repeated whenever the preselected job assignment proves to be infeasible. Hence, in the worst case, the complexity of the scheduling is increased at least by a factor proportional to the number of jobs.

While the focus of this thesis is on heuristics for performing job selection, preliminary evaluations of Heuristics A, B, and C verified that Heuristic B was the slowest, followed by Heuristic C and then A. Results also showed that Heuristic C produced by far the best results, when the goal was to minimize the completion time of the last job scheduled, with only a minimal computational time penalty as compared to Heuristic A. Heuristic C was therefore chosen as the heuristic used to investigate heuristics for performing job selection.

It must be noted that with constraints as complex as those in the crew activity planning problem, there exist problem instances with feasible solutions, but for which the above heuristics will not produce feasible schedules, regardless of the heuristic used for performing job selection. Consider, for example, using Heuristic C to solve a problem with four jobs, all requiring one crewmember to perform them (Figure 5.2). Suppose there are two crewmembers available, and both crewmembers have identical processing times for each job. Let Job 1 and Job 2 require 2 time units each, and Job 3 and Job 4 require 1 time unit each. Also suppose that there are no resource constraints or time constraints, except for two concurrence constraints. The first requires that Job 1 and Job 3 start at the same time, and the second requires that Job 2 and Job 4 start concurrently. Clearly, a feasible schedule would occur if Job 1 and 2 were each assigned to the same crewmember and Job 3 and 4 were then scheduled to meet the concurrence constraints (Figure 5.2A).

If the two long jobs (Jobs 1 and 2) are the first two jobs selected (in either order), then Heuristic C will assign these jobs one to each crewmember. The partial schedule will schedule both of these two jobs starting at time zero. If an attempt is then made to schedule either of the short jobs (Jobs 3 and 4) it will prove fruitless, because it is impossible to meet the concurrence constraints (Figure 5.2B). A similar problem will develop if the two short jobs are first scheduled (Figure 5.2C).

Figure 5.2: Failure to Find a Feasible Solution



Suppose the first two jobs selected include a long job and a short job. Consider first the case where this is Job 1 and Job 4 (or similarly, Job 2 and Job 3) (Figure 5.2D). Both of these jobs will be scheduled to begin at time zero, and each will be assigned to different crewmembers. It will then be impossible to add either Job 2 or Job 3 to the schedule feasibly.

The final case is if the short and the long jobs picked are jobs linked by the concurrence constraints. This would occur if Job 1 and Job 3 were the ones selected. Again, both of these jobs will be scheduled to begin at time zero, and each will be assigned to different crewmembers. When Job 2 or Job 4 is then selected for scheduling, it will be assigned to the same crewmember as Job 3, and it will begin at one time unit into the schedule (Figures 5.2E and F). As Job 1 will still be in progress at this time, it will be impossible to meet the remaining concurrence constraint in scheduling the last job. A similar situation would develop if Job 2 and Job 4 were the first two jobs selected for scheduling.

It is therefore seen that for this problem, Heuristic C will never produce a feasible schedule, even though one is possible. Heuristic A will also not produce a feasible schedule. It is possible that Heuristic B will produce a feasible solution, but only with a proper pre-planning of crewmember assignment and a job selection algorithm which will order the jobs in a fortuitous manner.

There are therefore problems for which there does exist feasible solutions but for which the algorithms used in this thesis will fail to even produce feasible schedules. This type of problem is likeliest to arise only when there are jobs whose start times are rigidly tied (by time constraints) to the start time of other jobs. For example, the algorithms will never fail to produce feasible schedules in the resource constrained multi-project scheduling problems of Section 4.3.1. The algorithms' lack of complete robustness is not surprising, because the general scheduling problem is NP-complete (Section 4.1) and hence no polynomial algorithm can guarantee finding even a feasible schedule.

The failure of the algorithm to work in all cases is not a crippling problem. In the testing of the algorithm (see Section 5.2), a feasible solution was always found if one was known to exist. The only times the algorithm was shown to fail was in problems, such as the one above, which were deliberately contrived to demonstrate this defect.

5.1.3 The Maximum Compatibility Method

The maximum compatibility method is motivated by the observation that good schedules usually have significant "overlap" between jobs. Jobs are said to overlap if they are performed at the same time (at least in part). If a method could be devised by which compatible jobs (i.e., jobs which can overlap, at least in part) were scheduled after each other, than one could expect to find significant overlap in a resulting schedule.

Key to this result is the assembly and use of the compatibility matrix, which must be designed to accurately reflect the desirability of each job being dispatched after each other job. If there are n jobs to be scheduled, the the compatibility matrix consists of an n x n array of numbers. The "compatibility" of Job A being dispatched after Job B is indicated by the entry in row A, column B of this compatibility matrix. Given this matrix of "compatibilities" (whose derivation will be described below), the rating of a pathway (dispatch ordering, or sequencing) through the compatibility matrix is determined by summing the values in the matrix corresponding to each pair of jobs in the pathway. For example, if a sequencing designated the scheduling order to be Job 3, then Job 1, then Job 2, then the rating of the pathway would be the sum of the element of the matrix in row 3, column 1, and the element in the matrix in row 1, column 2. Early serial versions of this algorithm attempted to predetermine the sequencing of the jobs by finding a pathway (ordering) through the compatibility matrix of the highest total rating. This problem, however, is similar to the well known traveling salesman problem, which is NP-Hard. However, through empirical testing, it was found that: 1) pathways with high ratings produced significantly better scheduling results than pathways with low ratings; 2) randomly produced pathways had a high probability of having low ratings; and 3) when differences in the ratings of orderings were small, there was no significant correletion between ordering ratings and resultant schedule quality. It was therefore found that there was no real advantage to finding the highest rated pathway; any highly rated pathway would likely produce a good schedule, and even better results could be obtained by searching through many different highly rated pathways and then keeping the best resulting schedule.

Many simple heuristics can be used for finding a highly rated pathway. An obvious method is to start with the two jobs corresponding to the highest entry in the matrix. Suppose, for example, that this entry was in row 7, column 11. The first two jobs to be selected for scheduling would then be Job 7 followed by Job 11. One would then go to row 11 of the matrix and select the largest entry corresponding to an unselected job. This procedure would then be repeated until all the jobs had been added to the ordering.

It was then found that even better results could be obtained by making the algorithm a parallel algorithm. The next job scheduled at each point would be selected by examining the row corresponding to the last job completed in the current partial schedule, which is not necessarily the last job scheduled. For example, suppose that Job 7 is dispatched, followed by Job 11. If Job 11 is scheduled so that it is completed before Job 7 is completed, then the algorithm would again select the next job for scheduling from the row corresponding to Job 7.

Other variations of the algorithm were also tried, such as examining the rows corresponding to the last two jobs in the current partial schedule, and combining their ratings scaled to 25% from the second to last job completed and 75% from the last job completed. The rationale for this was that, in some schedules, it would be common to have several (more than two) jobs overlapping at once, and that this would tend to select jobs which were compatible with the last several jobs. In practice, results obtained with this method were not as good as obtained by just looking at the last job completed (although results obtained were still considered quite good). It is anticipated that for some problems this approach will prove fruitful.

The compatibility matrix is designed to reflect both the ability of job pairs to overlap and the difficulty of scheduling particular jobs. For example, if a particular job is only capable of overlapping with a small number of jobs, then it is important to heavily rate those compatibility matrix entries so that it is likely that this job is selected before and/or after jobs for which it is compatible. Conversely, it would not be desired to have highly flexible jobs (which can overlap with many other jobs) scheduled before or after each other, because this would "waste" their ability to overlap.

The creation of a methodology to generate the compatibility matrix contains many options. Many methods were tried and a synthesis was achieved to determine what works and what does not work. The following steps, and the motivation for them, were the ones finally used to generate the compatibility matrix:

First, all job pairs were examined to determine the extent in which the jobs could overlap. The degree to which two jobs can overlap is determined by computing how much earlier (in time units) the two jobs could be completed by being dispatched one after the other (ignoring the effects of any other jobs). A worked example of computing a compatibility matrix is provided in Section 5.2.4.

Jobs cannot overlap if their simultaneous execution would exceed any resource limits. It is possible, however, for two jobs to overlap if the number of crewmembers they require exceeds the total number available, because some of the crewmembers may complete a job earlier than others. For example, suppose there are five crewmembers with completion times 15, 20, 20, 30, and 40 for Job 1, which requires 3 crewmembers to complete it. In addition, suppose that Job 2 also requires 3 crewmembers and can be completed by each of the crewmembers with a completion time of 25. If resource levels are not exceeded, it would be possible to schedule Job 2 after Job 1 with a overlap of 5 time units. This would occur by having Crewmembers 1, 2, and 3 perform Job 1, while Crewmembers 1, 4, and 5 would perform Job 2. Crewmember 1 would thus be working on Job 2 while Crewmembers 2 and 3 were finishing Job 1. It should also be noted that no overlap is possible for the case where Job 1 follows Job 2.

Jobs can also be prohibited from overlapping because of time constraints which interrelate them. For example, if two jobs are linked by a precedence constraint, then obviously they cannot overlap. Constraints specifying a minimum interval between the start of two jobs can also prohibit or limit the ability of jobs to overlap.

Once values are obtained for the number of time units with which job pairs can overlap, they are assembled into a matrix, with the row number representing the preceding job in the job pair, and the column number representing the following job in the job pair. The diagonal elements of this matrix are set to equal negative 1 (no job can follow itself), as well as those corresponding to any job pairs for which the time constraints prohibit they follow each other. For example, if a precedence constraint states that Job 2 must follow Job 1, then the entry in the first row, second column of the matrix would be zero (because while Job 2 cannot overlap with Job 1, it can follow it), while the entry in the second row, first column would be negative one, indicating that this ordering is prohibited.

When this matrix is assembled, the various rows and columns are multiplied by a factor representing the difficulty of scheduling each job with overlap. For each job j, a number f_j is determined from the matrix which is the minimum of: 1) the number of jobs with which it can overlap by preceding; and 2) the number of jobs with which it can overlap by following. This is just the minimum of the number of positive entries in row j of the matrix and in column j of the matrix. After all the f_j are determined, all the non-negative entries in each row j of the matrix are then multiplied by $1 + (\max f_j) - f_j$. Each of the non-negative entries in each column j of the matrix are also multiplied again by this same factor.

Some additional processing is performed to scale the matrix and remove the zero entries, so that the iterative techniques (described in Section 5.2.4.1) can be used. The largest entry in the matrix is found, and all entries in the matrix greater than zero are multiplied by a constant so that this largest entry is scaled to 99. Each non-negative entry in the matrix is then increased by one, thereby eliminating all zero entries from the matrix and making the largest entry equal to 100.

5.1.4 A Worked Example of the Maximum Compatibility Method

Consider a single resource scheduling problem with 5 jobs, 1 resource, and 3 crewmembers. Table 5.1 shows the number of crewmembers required for each job, the completion time of each job by each crewmember, and the resource usage of each job.

Table 5.1: Sample Scheduling Problem

Job	Resource Usage	# of Crew Required	Crew: 1	2	3 Performance Time
1	5	2	15	15	20
2	3	2	10	10	10
3	2	2	30	30	25
4	1	1	25	25	20
5	3	1	30	30	22

The resource limit for this schedule is 5 units. There is also a precedence constraint requiring that Job 1 precede Job 2.

STEP 1: Compute the minimum possible duration for each job. This is 15 for Job 1, 10 for Job 2, 30 for Job 3 (the second smallest completion time, as it requires 2 crewmembers), 20 for Job 4, and 22 for Job 5.

STEP 2: Start with row 1, and compute the overlap of each job if it follows Job 1 in the scheduling order. The first entry in row 1 will be -1 because Job 1 cannot overlap with itself. The rest of the entries in row 1 will be 0, because the resource constraint prohibits all of the other jobs from overlapping with Job 1. Also note that Job 2 cannot overlap with Job 1 because of the precedence constraint.

For row 2, compute the overlap of each job if it follows Job 2. The first entry in this row is -1 because this ordering is prohibited by the precedence constraint. The second entry in this row is also -1 because Job 2 cannot overlap with itself. The third entry in this row will be 0 because Job 3 cannot be started until Job 2 is completed, because too many crewmembers would be required. The fourth entry in row 2 will be 10. Consider that Crewmembers 1 and 2 are assigned Job 2 and Crewmember 3 is assigned Job 4. The jobs will then be completed in 20 time units, which is 10 time units less than the sum of the minimum completion times of Jobs 2 and 4. Finally, the last entry in the 2nd row will be 0 because the resource limit would be exceeded if the two jobs were to overlap.

For row 3, compute the overlap of each job if it follows Job 3. The first entry in this row will be 0, because the resource limit would be exceeded if Job 1 were to overlap with Job 3. The second entry will be 5. Consider that Job 3 is assigned to Crewmembers 1 and 3 while Job 2 is assigned to Crewmembers 2 and 3. Job 2 could then start at time 25, with a total duration for both jobs of 35, which is 5 less than 40, the sum of their individual minimum performance times. The third entry in this row will be -1, and the fourth entry will be 20. This would occur if because Job 4 can occur entirely within the duration of Job 3. This would have a total completion time of 30, which is 20 less than the sum of the minimum completion times of Jobs 2 and 4. Lastly, the fifth entry in this row is 22 because these two jobs can also competely overlap.

For row 4, the first entry is zero because of the resource limit. The second entry is 10, because Job 2 can be performed completely during the duration of Job 4. The third entry is 20 because Job 4 can be performed entirely within the duration of Job 3. The fourth entry is -1, and the fifth entry is 17, with would occur if Crewmember 1 or 2 performed Job 4 while at the same start time Crewmember 3 performed Job 5. This would have a net duration of 25, which is 17 less than the sum of the minimum durations of the individual jobs.

For the 5th row, the first two entries will be zero because of the resource limit. The third entry is 22, the fourth entry is 17, and the last entry is -1.

The matrix now has the form:

-1	0	0	0	0
-1	-1	0	10	0
0	5	-1	20	22
0	10	20	-1	17
0	0	22	17	-1

STEP 3: Compute Number of Positive Entries in Each Row = 0 1 3 3 2

Compute Number of Positive Entries in Each Column = 0 2 2 3 2

Compute Minimum (for each job) of these numbers = 0 1 2 3 2 = f_j

Compute Maximum of the f_j = 3

Compute 1 + (max f_j) - f_j = 4 3 2 1 2

STEP 4: Multiple Each Non-Negative Entry in Each Row and Each Column by these Numbers.

The Matrix now becomes:

-1	0	0	0	0
-1	-1	0	30	0
0	30	-1	40	88
0	30	40	-1	34
0	0	88	34	-1

STEP 5: Scaling the Matrix

The largest entry in the matrix is 88, so each positive entry is multiplies by 99/88.

The Matrix is now:

-1	0	0	0	0
-1	-1	0	33.75	0
0	33.75	-1	45	99
0	33.75	45	-1	38.25
0	0	99	38.25	-1

STEP 6: Add 1 to all non-negative entries

The Final Matrix is:

-1	1	1	1	1
-1	-1	1	34.75	1
1	34.75	-1	46	100
1	34.75	46	-1	39.25
1	1	100	39.25	-1

With the final compatibility matrix assembled, the heuristic can be applied. The first jobs selected for dispatching will be Job 3 and Job 5, because the matrix element at row 3, column 5 is the largest (or alternately row 5, column 3). After Job 5 is selected, Job 4 is selected next because the largest entry in row 5 corresponding to an unscheduled pending job is in column 4. Job 1 will be scheduled next (Job 2 is not pending until Job 1 is dispatched). Finally, Job 2 will be the last job scheduled.

5.2 Use of Heuristics

5.2.1 Use of Multiple Heuristics

Use of this method merely requires using all the heuristics to find schedules, and then taking the best resulting schedule.

5.2.2 Randomization of Ratings

The heuristics can employ the sampling method (Section 4.4.2) so that the (prioritized) ratings determined by the heuristics indicate relative probabilities that the jobs will be selected. For heuristics which try to find the job with some maximum value, this is straightforward, but for heuristics which attempt pick a job with some minimum value, some adjustment is necessary. The method employed in this thesis was to take the reciprocal of the values determined by the heuristic, and to let these indicate the relative probabilities that each of the jobs will be selected. In order to prevent division by zero, if the heuristic determines that some jobs have a zero rating (as is possible, for example, with Heuristic 8, Minimum Earliest Start Time), then selection is always from among these jobs, with the relative probabilities of selection being just the priorities of those jobs. As noted in Section 4.4.2, with the sampling method Heuristic 6 becomes a random search, with an equal probability of selecting any pending job.

In practical operation, the sampling method can be employed to find many different schedules using the same heuristic, and then taking the best schedule found.

5.2.3 Searching the Weighting Space

An attempt was made to develop a technique where, instead of simply taking the best result from several heuristics, the results from the heuristics could be weighted. For example, Heuristic 3, the Minimum Slack Method, and Heuristic 4, Greatest Resource Demand, each utilize information regarding different knowledge sources. Heuristic 3 concerns time constraints, while Heuristic 4 concerns resource usage. The ratings from these heuristics could be combined in the hope that together they might produce results superior to those obtained from using either of them separately. By combining results from even more heuristics, it could be hoped that even better results could be obtained.

Let h_{ij} denote the rating that Heuristic i gives for Job j at some point in the scheduling process. Let w_i denote the weighting given to Heuristic i. Let H_j denote the final rating given to Job j. Then $H_j = \sum_i w_i h_{ij}$, where S is the set of all heuristics being weighted. The w_i define a continuous space (the weighting space, of the same dimension as the number of heuristics being weighted) which can be searched for better solutions. In a sense, this technique can be considered a generalization of simply taking the best result from several heuristics (Section 5.2.1). Consider, for example, when one of the w_i is equal to one, and the rest are equal to zero.

In spite of the motivating arguments above, this method did not prove a viable search method. There are several reasons this occurred. Even though the weighting space is continuous, the objective function (schedule quality) is not. Most small changes in the weights produce no change in job dispatch order, and hence no change in schedule quality. Because of the discrete nature of the objective function, as well as this "plateau" effect, it is not possible to use traditional approaches, such as gradient search, to look for better solutions.

Other attempts were made to search the weighting space using other hill climbing techniques. For example, a large grid in the weighting space could be sampled, and a smoothing function employed to find regions of promise. This technique was not productive because it was found that, while most solutions in the weighting space were better than random job orderings, solutions tended to vary considerably and without trend within the weighting space. It was concluded that there was no better method for finding the best solutions (within the weighting space) than by conducting a thorough search of the space.

Conducting a thorough search of the weighting space is not an attractive alternative. If the number of heuristics being combined is large, then conducting a thorough search of the weighting space is at least as complicated as directly searching for good job orderings. If, on the other hand, the number of heuristics being combined is small, then the number of different job orderings produced by the search will be small. These job orderings will also tend to be quite similar to each other, so they will not adequately span the search space to the original scheduling problem. The chance of finding a truly superior solution will then be small. It was thus found that searching the weighting space was not an attractive method for solving scheduling problems of this type. Heuristic 10 is, in fact, a combination of Heuristic 3 and Heuristic 9, but the weighing factors were chosen as 1, and were not varied.

5.2.4 Making Perturbations in Previous Schedules (Hill Climbing)

5.2.4.1 Iteration by Increased Prioritization

The basic heuristics of Section 5.1 can be modified by an iterative process, which successively examines schedules produced by the heuristics to attempt to identify which jobs are causing the schedule to "bottleneck," and then increasing the priorities of those jobs, thus increasing the likelihood that these jobs will be scheduled earlier on the next iteration. This process is modeled in the flowchart in Figure 5.3.

As the net rating of each job, and hence its dispatch ordering, is determined by multiplying the priority of the job by its heuristic weight, complications can arise when a job has a zero heuristic weight. This might occur, for example, when using Heuristic 4, Greatest Resource Demand. If a particular job uses no resources, then its net rating will always remain zero no matter how much its priority is increased. For heuristics of this type, it is necessary to add a small constant to each of the heuristic weights to therefore enable such jobs to be scheduled earlier.

As shown in Figure 5.3, it is necessary to decide if adding a job to a partial schedule will cause the schedule to become "worse" than the current best schedule. Measurements of schedule quality, such as the completion time of the last job scheduled or the sum of the completion times, are nondecreasing with the addition of each job and can thus be used to interactively gauge schedule quality. In the event that some metric of schedule quality is used which is not nondecreasing, then it will be necessary to modify the algorithm to schedule the jobs until either all jobs have been scheduled or some job is found which is not schedulable. If all jobs are scheduled, then a "critical job" will then have to be determined by some method which is based on the metric used to gauge schedule quality.

In order to use this method of iteration to find successive schedules, it is necessary to identify the "critical" and "bottleneck" jobs as indicated in Figure 5.4. The critical job is so termed because it is the job which causes the schedule to encounter difficulties; the critical job is either unschedulable or would cause the schedule to become worse than is acceptable. The solution to this problem would be to allow this job to be scheduled earlier in the next iteration of the scheduling process, thereby allowing it to possibly precede those jobs which were preventing it from being added to the schedule at an acceptable time.

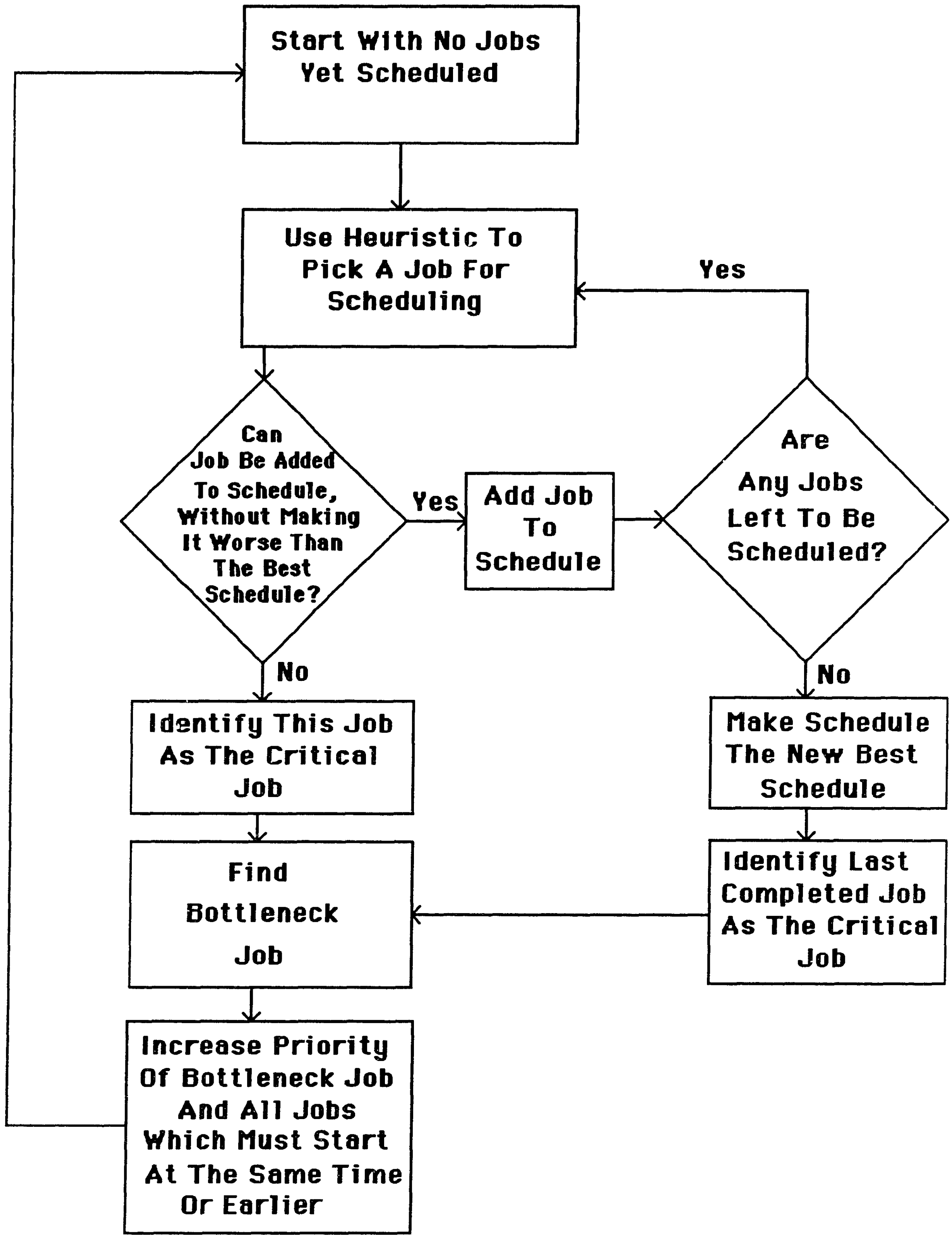


Figure 5.3: Flowchart of the Iterative Method

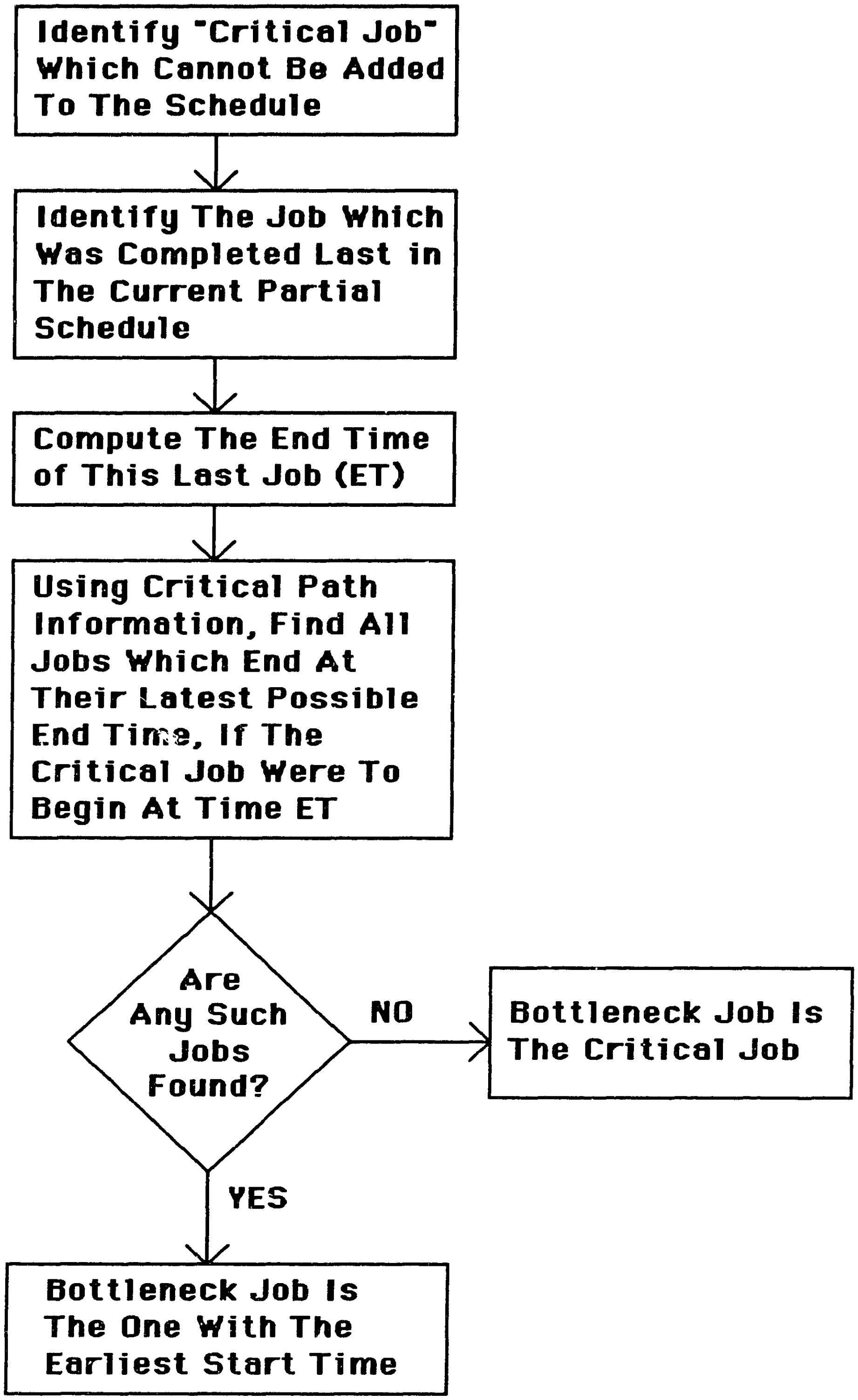


Figure 5.4: Identifying the Bottlenect Job

To accomplish this goal, it is not sufficient to simply increase the priority of this job. Consider a scheduling problem where there is some final job which cannot be done until all the other jobs have been completed. After generating an initial schedule, this final job would be the one identified as the critical job. This final job, however, is only pending once all the other jobs have been scheduled. Therefore, no matter how high its priority, it will never be scheduled before all the other jobs have been scheduled. Increasing the priority of this job would not be productive. What is necessary is that the priority of some of the jobs preceeding this final job be increased, thereby potentially allowing the formation of a better schedule, with the final job able to be scheduled at an earlier time.

Specifically, problems may occur if the critical job can not be scheduled (or scheduled earlier) unless other jobs, which are linked to the critical job by time constraints, are also scheduled earlier. For example, suppose that there is a precedence constraint requiring that Job 7 precede Job 11. Let the metric of schedule quality be to minimize the completion time of the last completed job. Suppose that Job 7 is scheduled, but if the scheduler were to attempt to schedule Job 11 directly after Job 7, it would have the job end later than the end time of the previous best schedule. It is would then be important to schedule Job 7 earlier, which could thereby permit Job 11 to also be scheduled earlier. In this case, Job 7 would be considered the "bottleneck" job.

In general, the bottleneck job is determined by the following steps:

1) Find the critical job, which cannot be added to the schedule.

2) Find the end time of the latest completed job which was successfully added to the schedule.

3) Utilizing critical path information (see Section 4.2), find the minimum permissible differences in start times between all scheduled jobs and the critical job.

4) Find all scheduled jobs which would be separated from the critical job by the minimum start time difference (found in step 3), were the critical job to be scheduled starting at the latest end time found in step 2.

5) If no jobs are found in step 4, then the bottleneck job is just the critical job. Otherwise, the scheduled job found in step 4 with the earliest start time is designated as the bottleneck job.

Not specified in Figure 5.3 is an indication of how much the priority of the bottleneck job is to be increased. Results indicate that good performance is achieved by increasing the job priorities by a randomly chosen factor between one and two. (If Heuristics 9 or 10 are used, then instead of increasing the job priorities, the row and column of the compatibility matrix corresponding to the job are each multiplied by the random factor.) A simple example will illustrate why a different random factor should be chosen at each iteration. Suppose that the prioritized version of Heuristic 6 is used to schedule jobs, so that jobs are chosen for scheduling solely by choosing the job with the greatest priority. Further, suppose that there are three jobs, Jobs 1, 2, and 3, which have priorities of 1.2, 1.1, and 1 respectively. The jobs are thus initially scheduled in the order 1-2-3. To obtain a better schedule (one completed earlier) the priority of Job 3 is then increased. Suppose a constant factor of 2 is used for the increase of priorities. The jobs will then have priorities 1.2, 1.1, and 2, and will be scheduled in the order 3-1-2. The next iteration will then order the jobs 2-3-1. Another iteration will return the jobs to their original order, 1-2-3; future iterations will continue this cycle indefinitely. Increasing the job priorities by a constant factor thus causes several potential orderings to be neglected, such as 1-3-2. If priorities are increased randomly, however, a search for better schedules will avoid this type of potentially unproductive cycling.

Figure 5.3 also indicates that, in addition to increasing the priority of the bottleneck job, all jobs which must precede the bottleneck job (from critical path considerations) should also be increased in priority. Consider the jobs linked by the precedence network shown in Figure 5.5, where Jobs 1 and 2 must precede Job 3, and Jobs 4 and 5 must follow Job 3. Suppose an initial schedule is completed in which the jobs are dispatched in the order 1-2-3-4-5, with Job 5 being completed last in the resulting schedule. If Job 5 is then identified as the bottleneck job, it's priority will be increased, so that is can be scheduled before Job 4. If the next iteration has Job 4 as the last completed job, then its priority will be increased. The ordering among Jobs 4 and 5 will then be continually reshuffled in an attempt to find better schedules. Suppose, however, schedule quality can only be improved by reordering the dispatching of Jobs 1 and 2. Merely increasing the priority of the last job will never affect the ordering of these jobs.

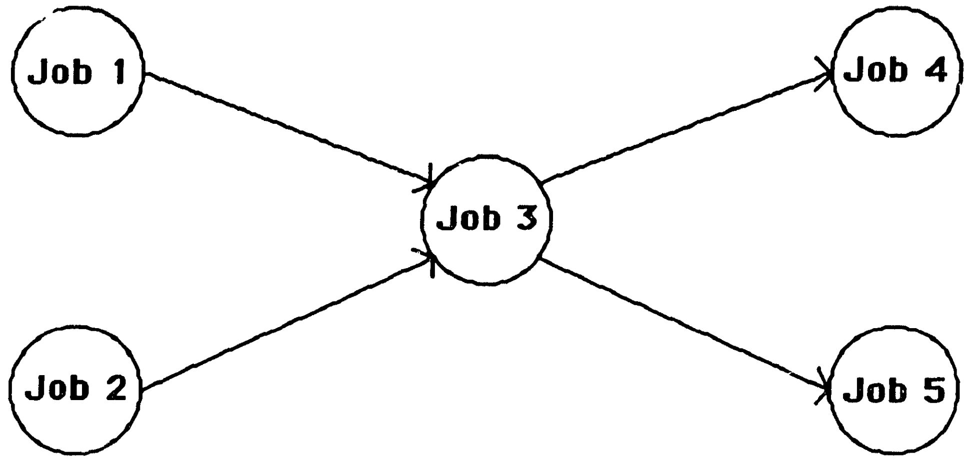


Figure 5.5: Five Job Precedence Network

A solution to this problem is to increase the priority of all jobs which must precede the bottleneck job, each by a different random factor. Thus whenever the priority of Job 4 (or Job 5) is increased, the priorities of Jobs 1, 2, and 3 are also increased, each by a different factor. This will thus allow rearrangement among the ordering of Jobs 1 and 2.

A particularly attractive feature of using increased prioritization for finding good schedules is that when the constraints become very complicated, initial execution of the heuristics in Section 5.2.1, or their randomized versions in Section 5.2.2, may not even produce a schedule which feasibly schedules all of the jobs. In typical operation, the iterative approach causes the jobs which are difficult to schedule to be scheduled earlier, while those jobs which are easy to schedule percolate toward the end of the ordering. The iterative algorithm is thus a method which intelligently allows those jobs causing difficulties to be reshuffled (hopefully towards feasibility) while still preserving use of the knowledge embodied in the heuristic. The iterative algorithm can therefore find feasible schedules where straight applications of the heuristics would fail. Even in scenarios where there are too many jobs to possibly be scheduled within the timeplan window, this method can be utilized to maximize the number of jobs scheduled by making the metric of schedule quality simply the total number of jobs included in the schedule. Increased prioritization was used by Grone and Mathis [Grone and Mathis, 1980] to force inclusion of "politically" important jobs which were given low ratings by their heuristic and were therefore not included in an initial schedule.

5.2.4.2 Randomization and Increased Prioritization

The sampling method can be used in conjunction with increased prioritization. When the priorities of the bottleneck job and its predecessors are increased, the probabilities of selecting these jobs are also increased on the next iteration.

5.2.4.3 Making a Schedule 2-Optimal

Attempts can be made to improve schedule quality by switching the ordering of jobs in a preliminary schedule. This technique has been used successfully for finding improved solutions to travelling salesman problems and in precedence constrained routing problems [Psaraftis, 1983]. In actual practice, however, this technique proved to suffer from several difficulties when applied to the crew activity planning problem. Most of these difficulties stem from the fact that there are on the order of n squared possible job pairs which can be switched, where n is the total number of jobs. In typical operation of this algorithm, one would systematically switch pairs of jobs in the ordering sent to the dispatcher. This would continue until either an improved schedule was found (in which case the algorithm would restart) or all possible job switches had been investigated (in which case the algorithm would terminate). In converging to a 2-optimal solution, in which no job switch will result in schedule improvement, it may therefore be necessary to restart the algorithm several times.

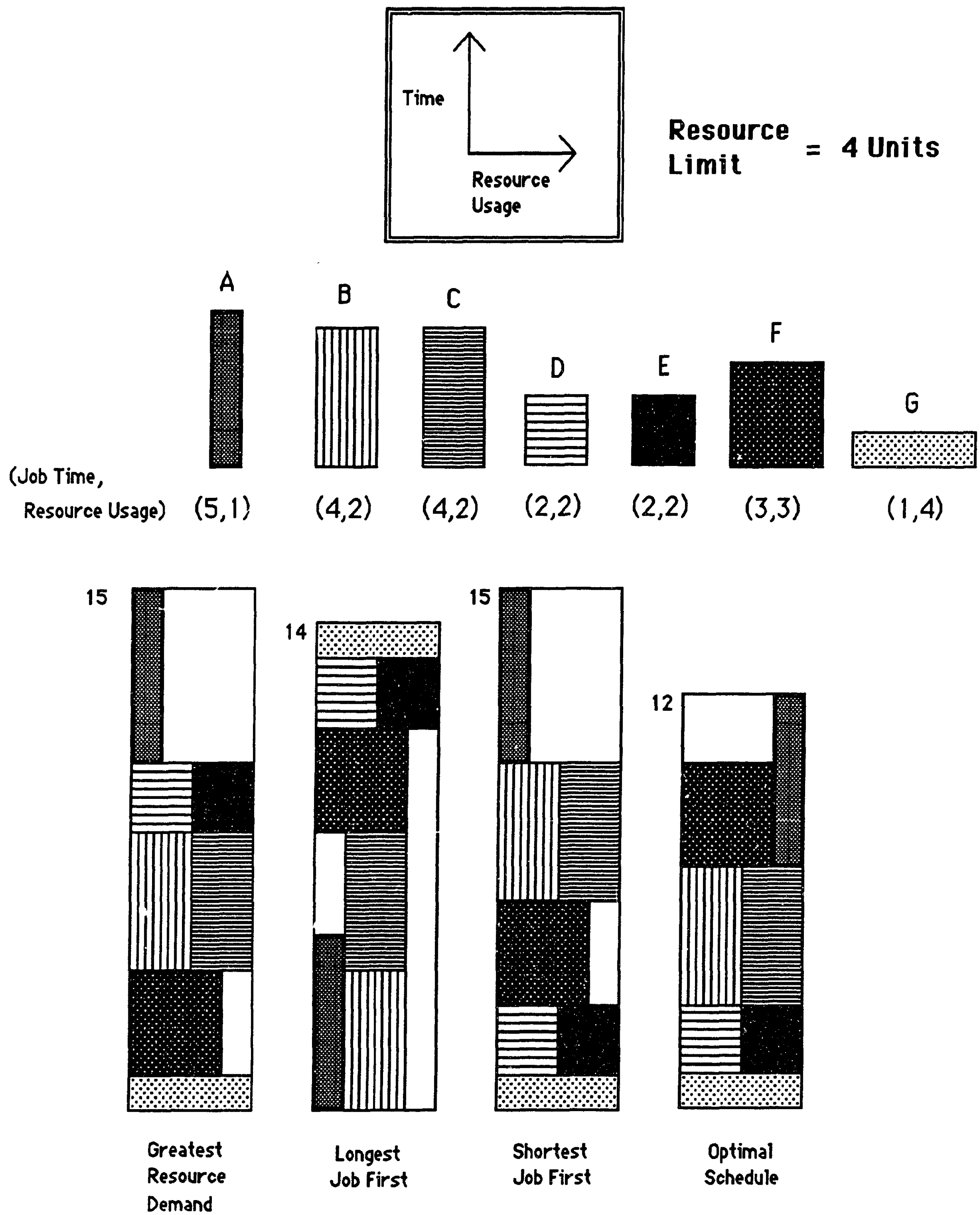
In scheduling problems where there are time constraints, job switching will often result in spending much time investigating infeasible job orderings. When no time constraints are present, preliminary emperical results indicate that job switching does result in schedule improvement, but not of the magnitude or dependability obtainable from increased prioritization. More significantly, computational time required for a schedule to converge to 2-optimality is much greater than that required from increased prioritization, due to the n-squared nature of the algorithm and the necessity for many restarts.

5.2.4.4 A Worked Example of Increased Prioritization

Figure 5.6 illustrates a scheduling problem consisting of seven jobs, labelled A through G. Each of these jobs has a duration (indicated by the height of the block corresponding to each job

Figure 5.6

Illustration Of Iterative Search Methodology



in Figure 5.6) and each job uses an amount of a single limited resources, which is indicated by the width of each block in Figure 5.6. The goal is to find a scheduling of these seven jobs which finishes as early as possible, subject to the constraint that no more than four units of the resource be used at any one time. This problem is equivalent to that of stacking the blocks into a box of width four so that the box is as short as possible.

Figure 5.6 shows how three different heuristics yield solutions of total duration 15, 14, and 15. Also shown in this figure is an optimal solution of duration 12. Figure 5.7 shows how the iterative algorithm is used with the Longest Job First Heuristic to find an optimal solution.

Initially, all the jobs have a priority of one. As scheduling order is determined by multiplying each jobs priority by its duration, the jobs are initially scheduled in order of decreasing job length {ABCFDEG}, with a total schedule duration of 14. As Job G has the latest completion time, its priority is then increased on the next (second) iteration by a random factor between one and two (in this case, 1.7). This is not sufficient, however, to move Job G ahead of Job D or E in the scheduling order. Therefore another iteration again multiplies the priority of Job G by another randomly generated factor of 1.4. The priority of Job G is now 2.38. This is enough to move Job G ahead of Jobs D and E, and this produces a new configuration {ABCFGDE}, as illustrated in the picture for Iteration 3. While this change produces a new configuration, it does not improve the duration of the solution to shorter than 14.

Jobs D and E are now the last jobs completed, so their priorities are now increase by random factors of 1.3 and 1.7 respectively. This yields a new ordering of the jobs {ABCEFDG}, still of duration 14. Job G is now again the last job scheduled, and its priority is increased by a factor of 1.1 for then next iteration (iteration 5). Iteration 5 thus orders the jobs {ABCEFGD}. While Job G is no longer the last job added to the schedule, it is still the last job completed, because there is an earlier open spot in which Job D can be scheduled but Job G cannot fit. For the next iteration, it is therefore Job G that has its priority increased (by a factor of 1.3) giving the ordering {ABCGEDF}, shown in the picture for iteration 6. Finally, a seventh iteration, increasing the priority of Job F by 1.6, gives an optimal configuration of duration 12, as shown in Figure 5.7.

Figure 5.8 shows the similar procedure applied to the Greatest Resource Demand Heuristic. For this heuristic, the process converges in only 3 iterations.

Figure 5.7: Longest Job First Heuristic

Random Numbers: 1.7, 1.4, 1.3, 1.7, 1.1, 1.3, 1.6

Job	Job Length	Iter. 2	Iter. 3	Iter.4	Iter. 5	Iter. 6	Iter. 7
A	5	5	5	5	5	5	5
B	4	4	4	4	4	4	4
C	4	4	4	4	4	4	4
D	2	2	2 *	2.6	2.6	2.6	2.6
E	2	2	2 *	3.4	3.4	3.4	3.4
F	3	3	3	3	3	3 *	4.8
G	1*	1.7 *	2.38	2.38 *	2.618 *	3.4034	3.4034
Ordering:	ABCFDEG	ABCFDEG	ABCFGDE	ABCEFGD	ABCEFGD	ABCGEDF	AFBCGED

An * indicates the job whose priority is increased on the next iteration.

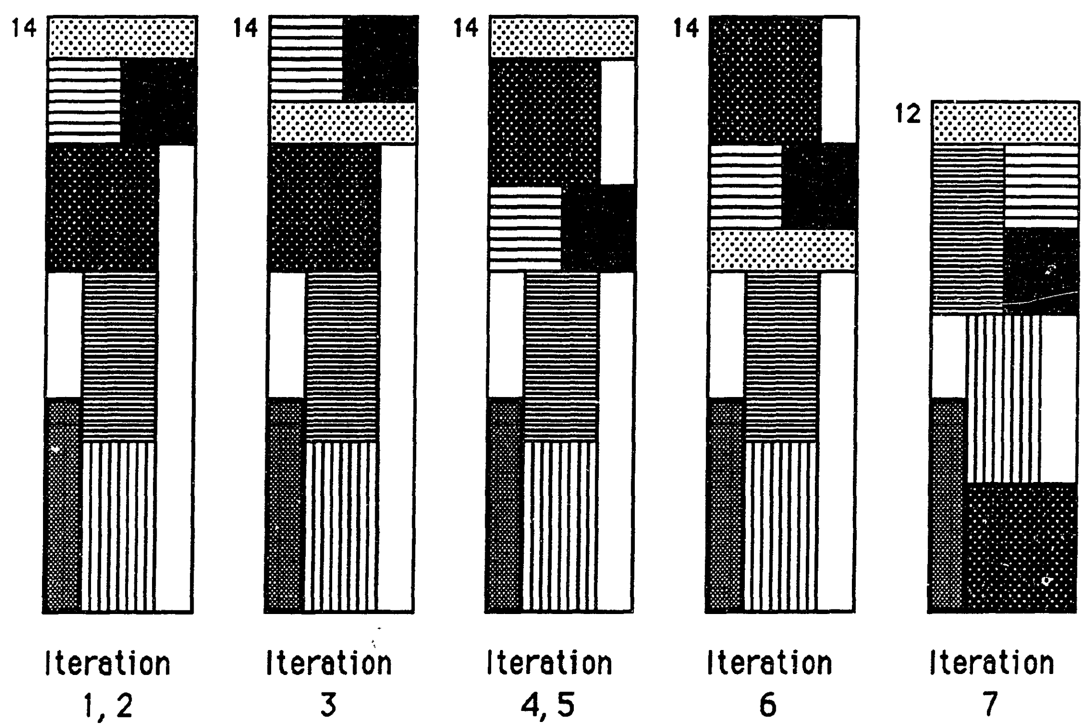
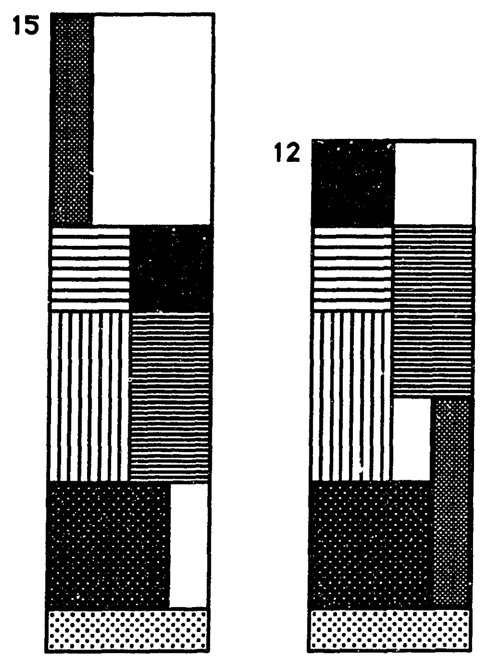


Figure 5.8: Greatest Resource Demand Heuristic

Random Numbers: 1.7, 1.4

Job	Resource Usage	Iter. 2	Iter. 3
A	1*	1.7*	2.38
B	2	2	2
C	2	2	2
D	2	2	2
E	2	2	2
F	3	3	3
G	4	4	4
Ordering:	GFBCDEA	GFBCDEA	GFABCDE

An * indicates the job whose priority is increased on the next iteration.



Iteration 1, 2   Iteration 3

While this problem illustrates the mechanics of the iterative algorithm, it must be remembered that applying the process to a robust scheduling problem is a much more complicated procedure, although it is still possible to extend the analogy of stacking blocks into a box to some extent. Time constraints, such as precedence constraints, indicate whether and by how much some block must be above or below others. Constraints such as earliest and latest start time constraints and target constraints may require that some blocks be restricted to certain segments of the box. Still other constraints, such as creating jobs with multiple resources or having job duration depend on which crewmember(s) perform it are further complications which are not as easily incorporated into a blocks model.

5.3 Results of Testing the Heuristics

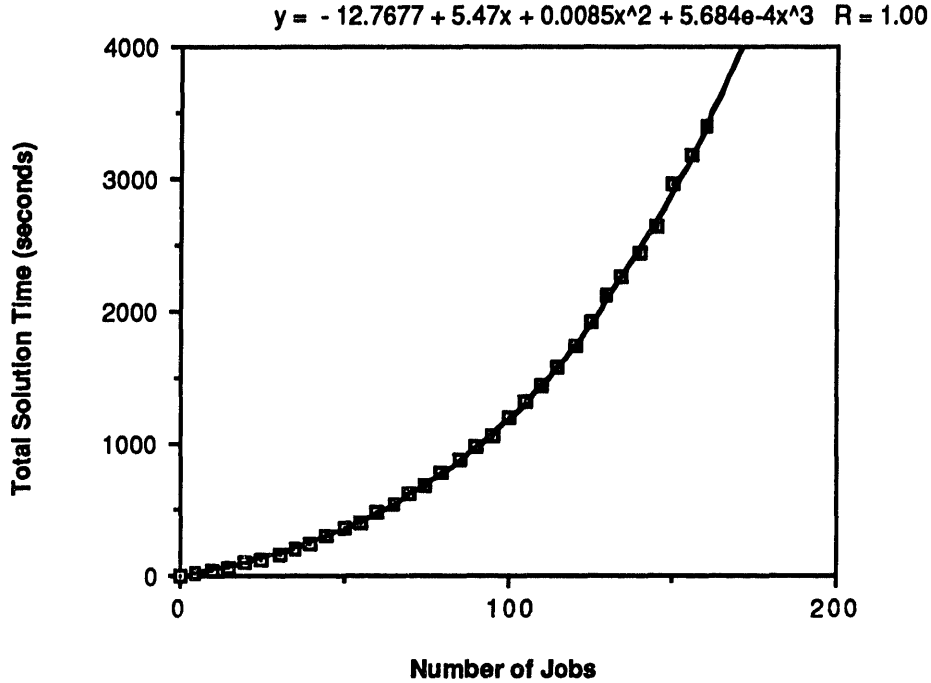
The complexity of the activity planning problem as well as the complexities of the heuristics preclude analytic analysis of heuristic quality. As has been the practice in the scheduling literature, it is therefore necessary to use monte carlo simulation over a range of problems in order to gauge the efficiencies of the heuristics [Davies, 1973; Davis, 1973a; Patterson, 1973; Davis and Patterson, 1975; Patterson, 1976; D. Cooper, 1976; Kurtulus and Davis, 1982]. Further, as many of the heuristics and algorithms involve the use of random variables, it can be necessary to execute many trials of the same heuristic technique on the same problem, in order to properly gauge the technique's expected efficiency and variability [D. Cooper, 1976].

Analysis of the ten heuristics of Section 5.1, utilizing the methodologies of Section 5.2, was performed, and is presented in this section. Appendix A presents the formulation of the test problems used to evaluate the heuristics. Also in Appendix A are results for each specific problem. This section uses those results to find answers to many questions which are critical to the determination of the relative qualities of the various heuristics.

5.3.1 Computational Complexity

The scheduling of each of the n jobs requires on the order of n squared steps, and, as there are n total jobs, the complexity of the algorithms used to implement the heuristics for scheduling are thus of the order of the cube of the number of jobs, as is shown in Figure 5.9. Figure 5.9 shows computational time versus number of jobs. To generate this data, a basic data base of five jobs and no time or target constraints was used. Larger problems were generated by making repeated copies of these five jobs. It is seen that the results conform well to an n cubed hypothesis, although the effect of the third order term (shown as $5.684 \times 10^{-4} \times X^3$, where X is the number of jobs) is small over the range of job sizes examined. (Note: The constants in the equations which describe the best fit curves for Figure 5.9 through Figure 5.18 are functions of the structure of the particular problem which was used for testing; other problems will have other constants for these curves, but the order of the curves will be unchanged. Following the equation describing the best fit curve in each of these figures is the equation R=1.00 showing that the correlation coefficient for each of these curves is 1, to an accuracy of two significant decimal places.)

Figure 5.9: Solution Time vs. Number of Jobs

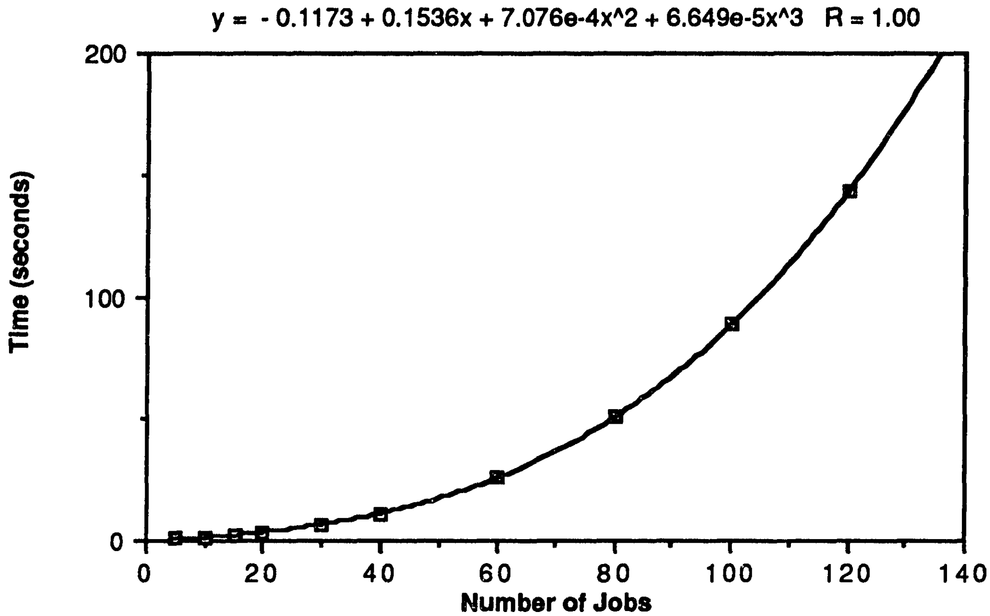


Each of the steps below breaks down the complexity of each step in the scheduling process, as it was implemented by the MFIVE Crew Activity Planner (Section 6).

STEP 1: Selecting a Job for Scheduling

Finding the pending jobs requires on the order of n squared computations (note: a more efficient implementation could probably reduce this to an order of n computations). Using the heuristic to compute a ranking value for all of the jobs requires on the order of n computations. Once the ranking values are computed, finding the job with the maximum or minimum ranking also requires on the order of n computations. Figure 5.10 shows computational time (for scheduling all n jobs) versus the number of jobs scheduled (i.e., n). The data conforms well to an n cubed model (i.e., n squared computations done n times).

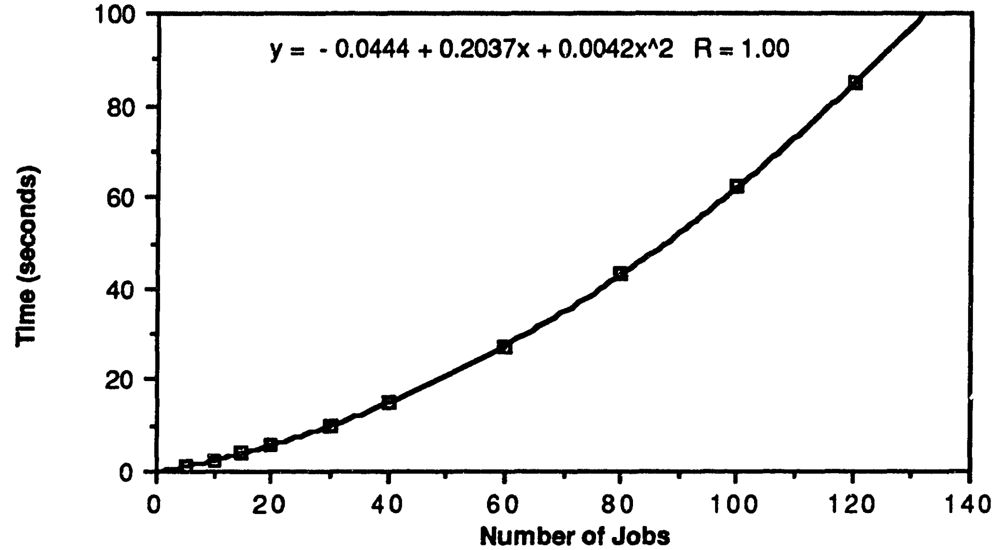
Figure 5.10: Time Required to Select the Dispatch Order



STEP 2: Applying the Time and Resource Constraints to the Job

Once a job has been selected for scheduling, its feasible windows for scheduling can be determined by: 1) examining its earliest and latest start times (as determined using the algorithms presented in Section 4.2); 2) examining its target constraints (if any) to find times during which the job is infeasible; and 3) examining the current partial schedule to find the time intervals during which the job can be performed without exceeding resource limits. This last step is the most complicated, because the length of the current partial schedule depends indirectly on the total number of jobs. This is because if there are n jobs, then there are at most 2n times when the resource levels can change. The total computation to apply the time constraints to all n jobs is thus proportional to n squared. The data in Figure 5.11 conforms to an n squared model.

Figure 5.11: Time to Apply Time and Resource Constraints



STEP 3: Selecting the Crewmember(s) to Perform the Job

As discussed in Section 5.1.2, using Heuristic C, each of the possible choices of crewmembers must be tried. If all jobs require only one crewmember, then the number of alternatives is of the order of the number of crewmembers. If jobs may require more than one crewmember, then the number of alternatives is exponential with the number of crewmembers available. The number of computations involved in this step does not, however, depend upon the total number of jobs, except indirectly. When checking to see if and when a crewmember or crewmembers are capable of performing a job, the current partial schedule must be searched for feasible times. The length of the current partial schedule is indirectly a fuction of the number of jobs. Figure 5.12 shows computational time (summed over all n jobs) for searching (through all the subsets of crewmembers) for the subset of crewmembers who will complete each job earliest. This does not include the total computational time for finding (for each subset of crewmembers) the earliest time is (if one exists) in which the job can be completed. This is shown in Figure 5.13. Both graphs conform to an n squared model.

Figure 5.12: Time to Search Possible Crew Assignments

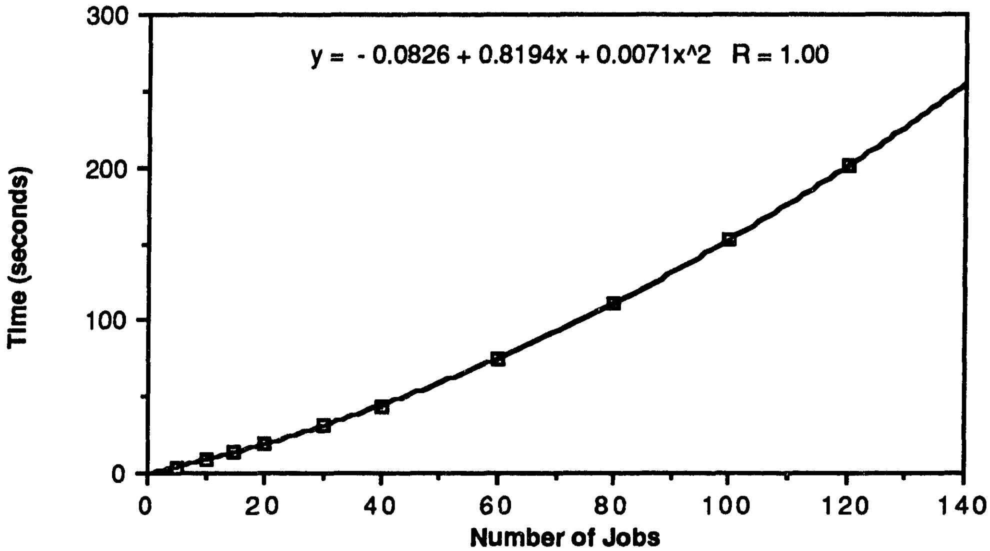
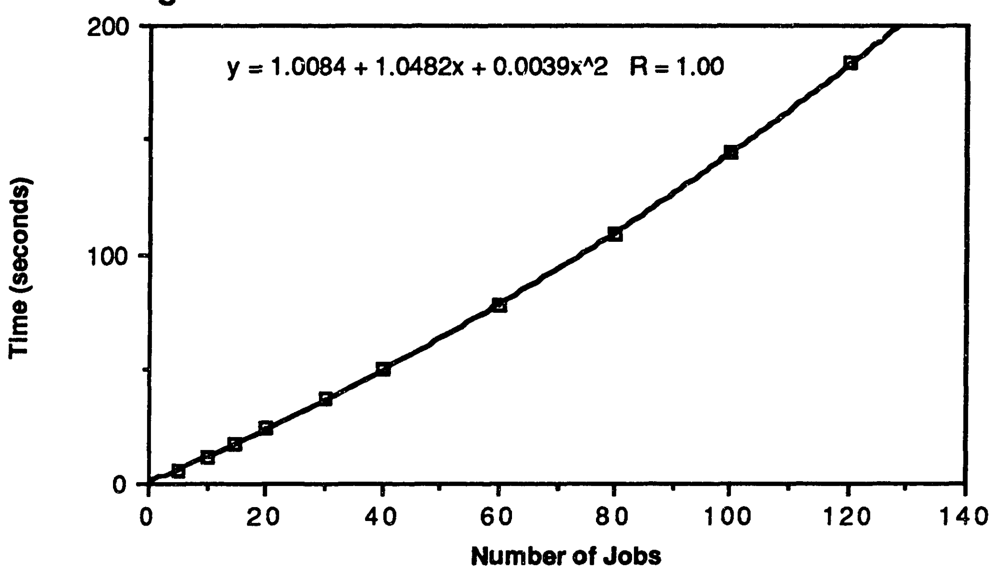


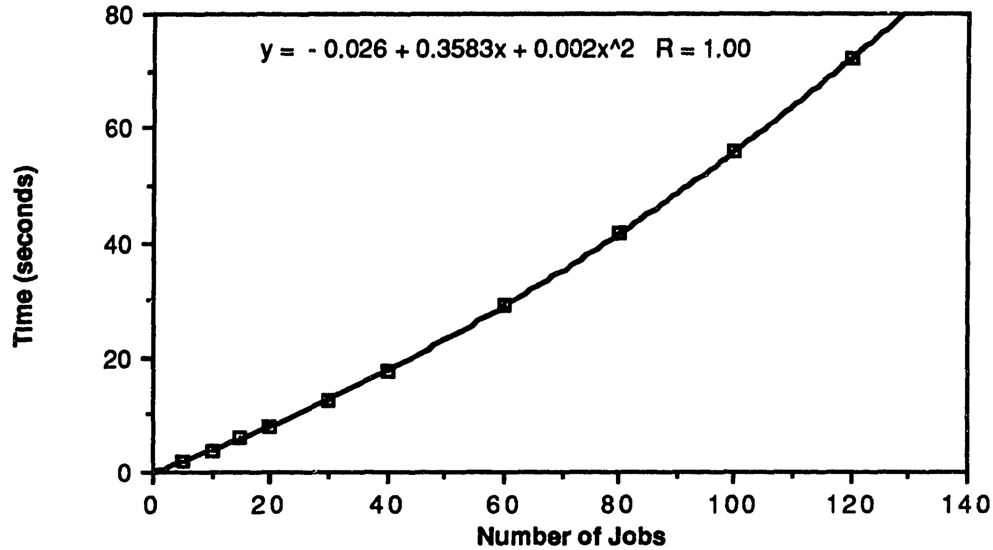
Figure 5.13: Time to Find Earliest Start Time for a Crew



STEP 4: Adding a Job to the Schedule

Adding a job to the schedule involves manipulation of the current partial schedule, whose length is proportional to the number of jobs. Figure 5.14 shows the total computational time for adding all n jobs to the schedule, and this graph conforms well to an n squared model.

Figure 5.14: Time Required to Add the Jobs to the Schedule



STEP 5: Propagating the Time Constraints

After adding the job to the schedule, active constraint propagation is performed in order to propagate the effects of scheduling this job on the other jobs. As described in Section 4.2.5, this operation requires computation on the order of the square of the number of jobs. This step is thus the "bottleneck" step in putting together the schedule. Figure 5.15 shows that this step conforms to an n cubed model.

In addition to the above five steps, which are performed for each of the n jobs, MFIVE uses computational time to update the screen display at the end of each iteration (Figure 5.16), and to manage (or oversee) the scheduling process (Figure 5.17). Computational time to update the screen display is proportional to the number of scheduled jobs which fit into the time window displayed on the screen. This computational time approachs a limiting constant as this window becomes full and any additional jobs are scheduled outside this window. The computational time to manage the scheduling process is constant for each job, and thus increases linearly with the total number of jobs. Table 5.2 shows the percent of the total computational time for each of the above areas for problems containing 15, 30, 60, and 120 jobs.

Figure 5.15: Time to Propagate Time Constraints

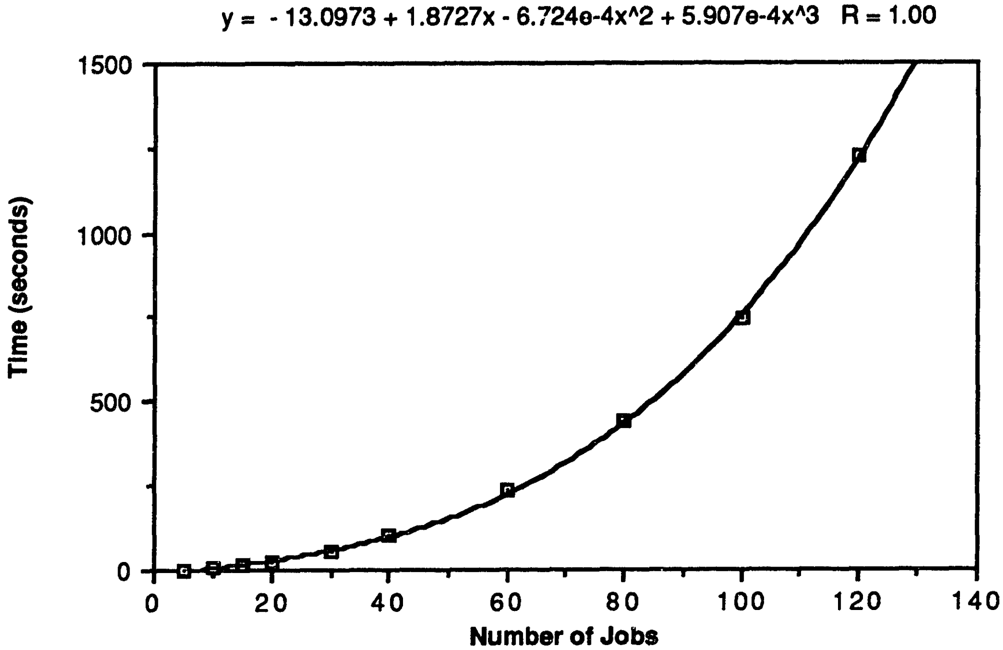


Figure 5.16: Time Required to Update Screen

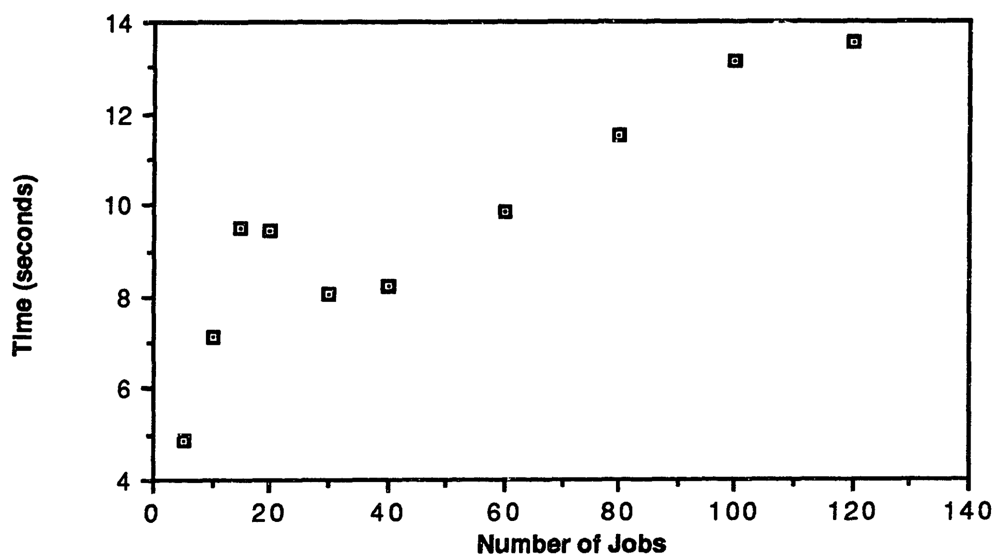


Figure 5.17: Time Required by "Master" Program

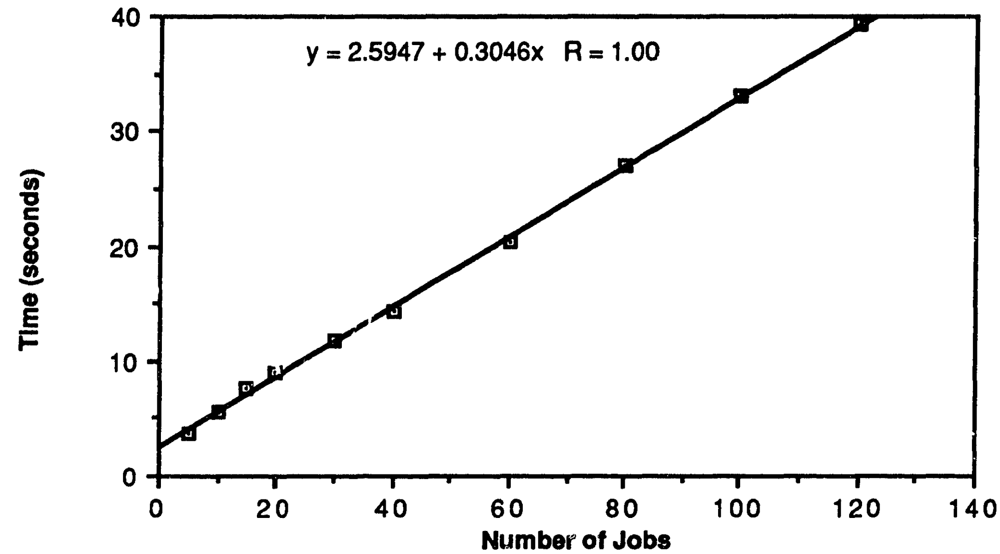
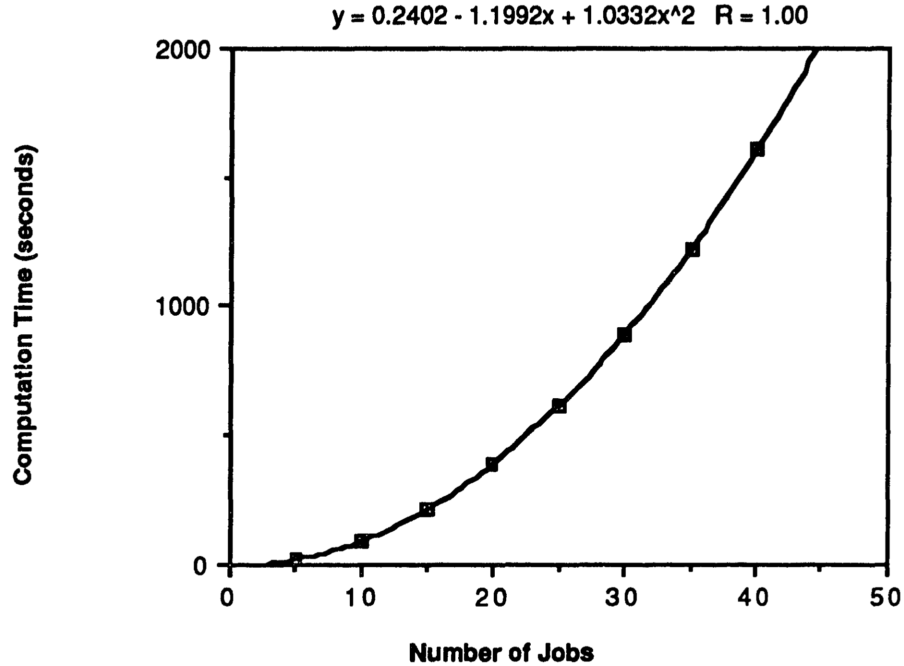


Table 5.2: Percent of Total Computational Time

Step in the Scheduling Process	15 Jobs	30 Jobs	60 Jobs	120 Jobs
Time Required to Select Dispatch Order	3.4	4	5.2	7.3
Time to Apply Time and Resource Constraints	5.2	5.7	5.4	4.3
Time to Search Possible Crew Assignments	23.3	21.9	15.6	9.4
Time to Find Earliest Start Time for a Crew	18.4	18	14.9	10.2
Time Required to Add the Jobs to the Schedule	7.8	7.2	5.7	3.6
Time to Propagate Time Constraints	18.6	31.5	47	62.4
Time Required to Update Screen	12.7	4.7	1.9	0.6
Time Required by "Master" Program	10.6	7.0	4.3	2.2
Total Time (seconds)	74.4	171.0	499.3	1965.0

Not mentioned in the above analysis is the setup cost associated with generating the compatibility matrix used for Heuristics 9 and 10. Generating this matrix requires a pairwise comparison of the jobs, and hence is proportional to the square of the number of jobs. Figure 5.18 shows computational time versus number of jobs for the generation of the compatibility matrix.

Figure 5.18: Computation Time vs. # of Jobs for the Compatibility Matrix



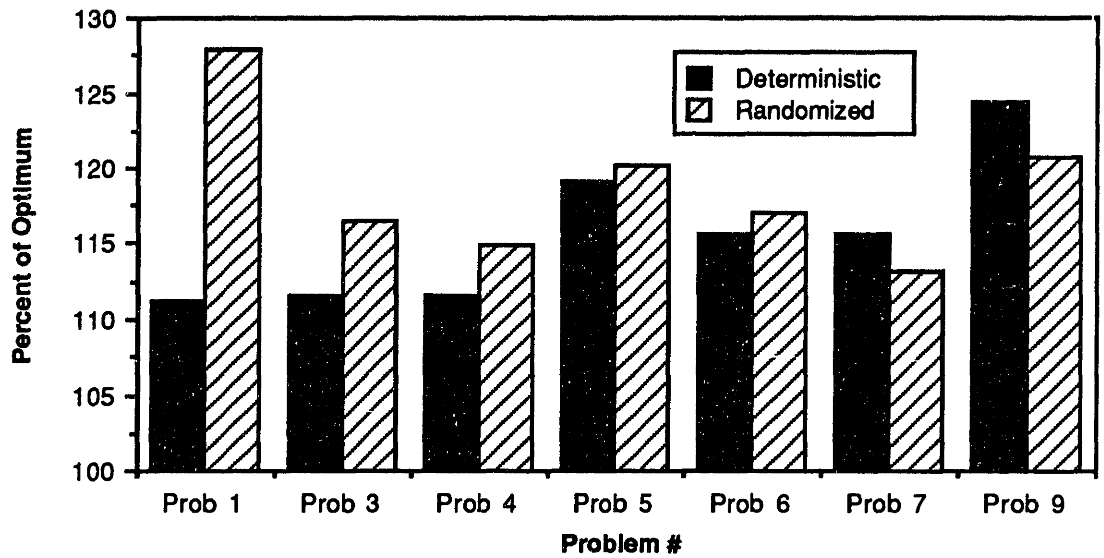
5.3.2 Initial Solutions

An attempt was made to determine whether the (non-iterated) deterministic performance of an heuristic was better than the average randomized (i.e., using the sampling method) performance of the same heuristic (again, non-iterated). Table 5.3 and Figure 5.19 show the averaged (over each of the 10 heuristics) results of applying the deterministic and randomized heuristics to 7 problems, expressed as percent of optimum. In order to obtain the value for the randomized heuristics, 100 trials were performed, and the results averaged (except for Heuristic 6 as applied to Problems 6, 7, and 9, in which case 1000 trials were performed, and the results averaged).

Table 5.3: Deterministic vs. Randomized Initial Solutions

Problem	Average Deterministic Solution	Average Randomized Solution
1	111.25	127.875
3	111.538	116.474
4	111.5	114.9
5	119.143	120.109
6	115.581	117.009
7	115.625	113.214
9	124.336	120.718
Average	115.568	118.616

Figure 5.19: Initial Solutions



As can be seen from this data, the deterministic solution averaged better for 5 of the 7 problems, about 3 percent better overall. Table 5.4 shows hows the two methods compared for each heuristic, again averaged over the same 7 problems.

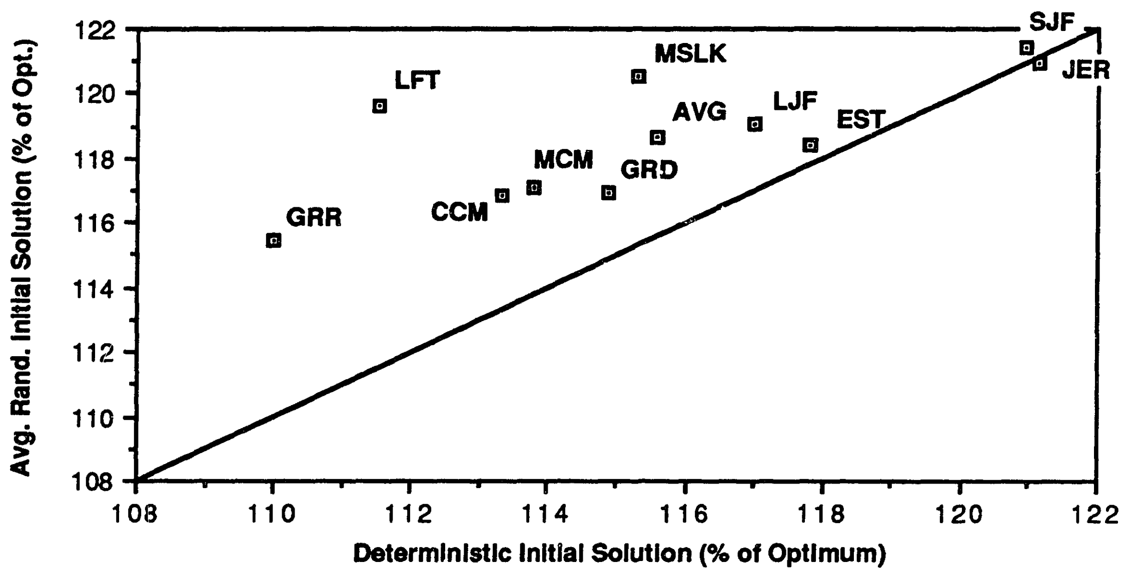
Table 5.4: Deterministic vs. Randomized Initial Solutions

Heuristic	Average Deterministic Solution	Average Randomized Solution
1 SJF	120.936	121.411
2 LJF	116.983	119.016
3 MSLK	115.322	120.561
4 GRD	114.883	116.893
5 GRR	109.992	115.441
6 JER	121.142	120.899
7 LFT	111.514	119.590
8 EST	117.822	118.413
9 MCM	113.780	117.066
10 CCM	113.305	116.843
Average	115.568	118.616

This data is shown graphically in Figure 5.20. As can be seen in Figure 5.20, the deterministic initial solutions averaged better for all heuristics except for Heuristic 6, **JER**, for which both methods were approximately equivalent (as would be expected). It can also be seen from this graph that there is a correlation between the quality of solution when an heuristic is applied in the standard deterministic fashion and the average quality of solution when it is applied in the randomized fashion.

Several other conclusions can be drawn from the above data. The first is that none of the heuristics fared significantly worse than the randomized version of Heuristic 6 (i.e., random scheduling), which was about 20% worse than optimum. Heuristics 5, 7, 9, and 10 were the best of the deterministic heuristics, Heuristic 5 averaging 10 percent better than optimum, a significant improvement over random scheduling.

Figure 5.20: Average Heuristic Performance



5.3.3 Final Solutions

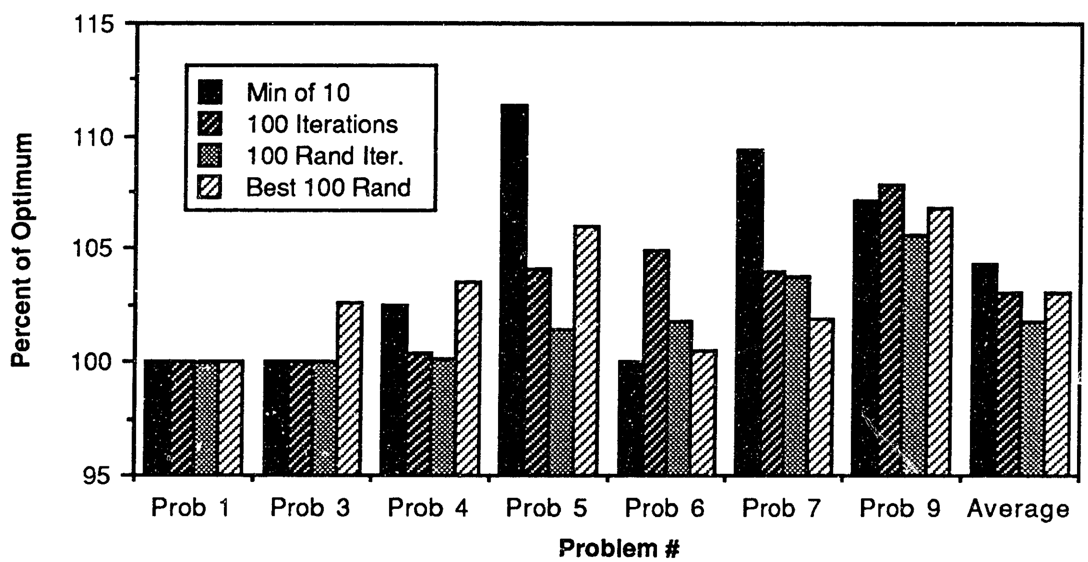
This section compares various searching techniques for finding improved solutions. Four techniques are examined: 1) applying all 10 heuristics once, and taking the best resulting solution; 2) performing 100 iterations of the deterministic versions of each of the heuristics, using the iterative algorithm described in this thesis; 3) performing 100 iterations of the each of the randomized versions of each of the heuristics; and 4) performing 100 applications of the randomized versions of each of the heuristics (without iterating) and taking the best resulting solution. Table 5.5 and Figure 5.21 compare each of these methods for several of the tested problems.

Table 5.5: Comparison of Techniques for Schedule Improvement

Problem	Min of 10	100 Iterations	100 Rand. Iterations	Best of 100 Rand.
1	100	100	100	100
3	100	100	100	102.564
4	102.5	100.42	100.167	103.5
5	111.429	104.095	101.429	106
6	100	104.884	101.784	100.465
7	109.375	104.011	103.75	101.875
9	107.163	107.884	105.654	106.781
10		107.188	106.541	
Average	104.352	103.042*	101.827*	103.026

*Not including data from Problem 10.

Figure 5.21: Finding an Improved Solution



The first and fourth technique are not applicable to Problem 10, for which they fail to produce complete solutions.

The results show that taking 100 iterations of the randomized heurisitic is the preferred technique. It worked best overall in 5 of the 7 problems. It is also capable of being applied to problems such as Problem 10 for which the fourth technique is not applicable. Table 5.6, Figure 5.22, and Figure 5.23 compare the deterministic and randomized iterative echniques compared for each heuristic, averaged over Problems 5, 6, 7, 9, and 10.

Table 5.6: Comparison of Deterministic and Randomized Iteration Techniques

Heuristic	100 Iterations (Deterministic)	100 Iterations (Randomized)
1 SJF	104.833	104.531
2 LJF	105.667	103.565
3 MSLK	105.836	104.819
4 GRD	107.017	102.770
5 GRR	107.190	103.996
6 JER	105.268	104.393
7 LFT	106.036	103.763
8 EST	106.617	104.356
9 MCM	103.865	103.288
10 CCM	103.792	102.836
Average	105.612	103.832

Figure 5.22: Deterministic vs. Randomized Final Solutions

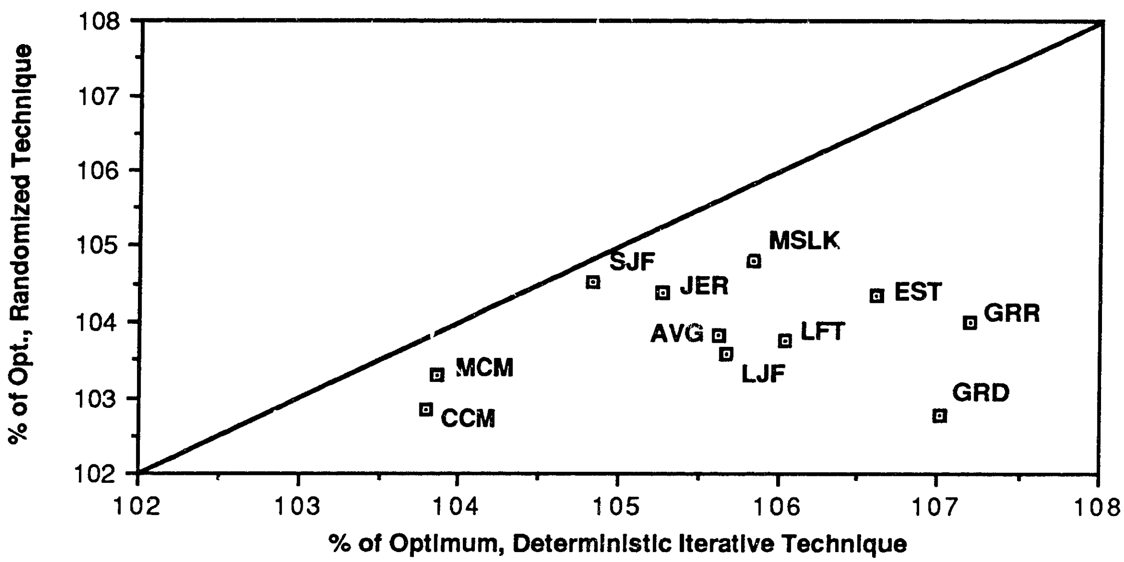
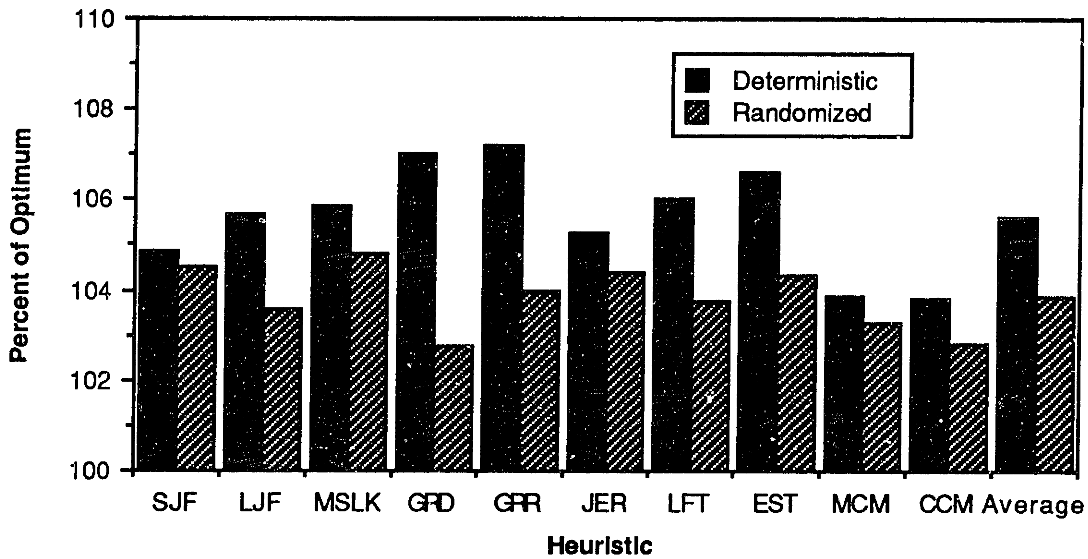


Figure 5.23: Deterministic and Randomized Final Solution

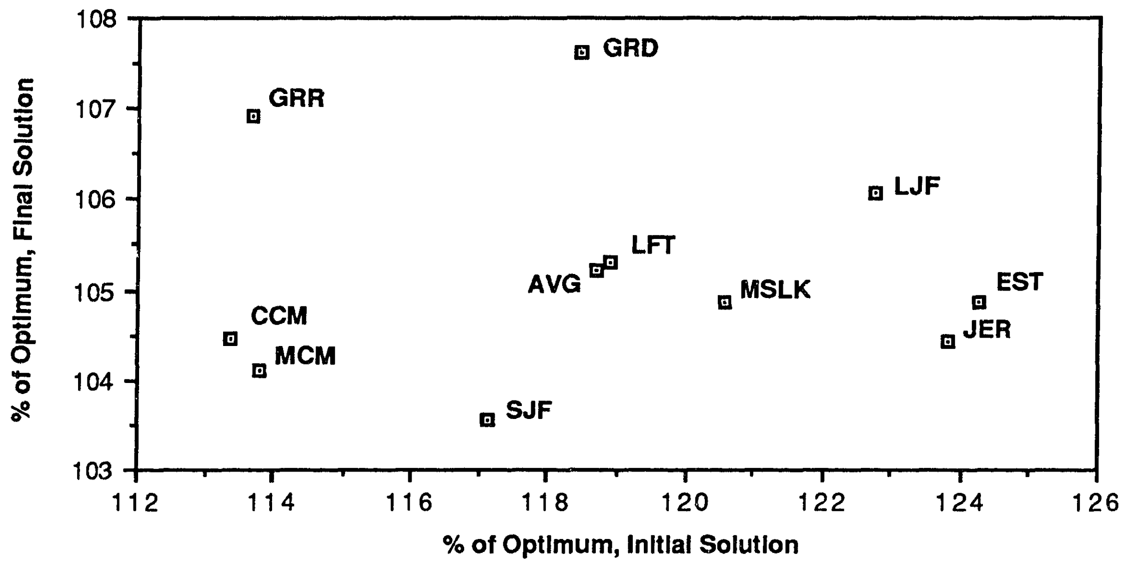


Without exception, the randomized versions of the heuristics produce superior solutions. Only three of the deterministic techniques, namely Heuristics 1, 9 and 10, produced results superior to Heuristic 6, the Jobs Equally Rated Heuristic. As Figure 5.22 shows, there is no strong correlation between the quality of the final solutions with the deterministic and randomized techniques.

5.3.4 Initial and Final Solutions

An attempt was made to determine whether heuristics which produce good (non-iterated) initial solutions are also likely to produce good final solutions after 100 iterations. Figure 5.24 compares the initial and final solutions for the deterministic versions of each heuristic, averaged over Problems 5, 6, 7, and 9. As is clear from the figure, there is no strong correlation between initial and final solutions.

Figure 5.24: Initial vs. Final Solution



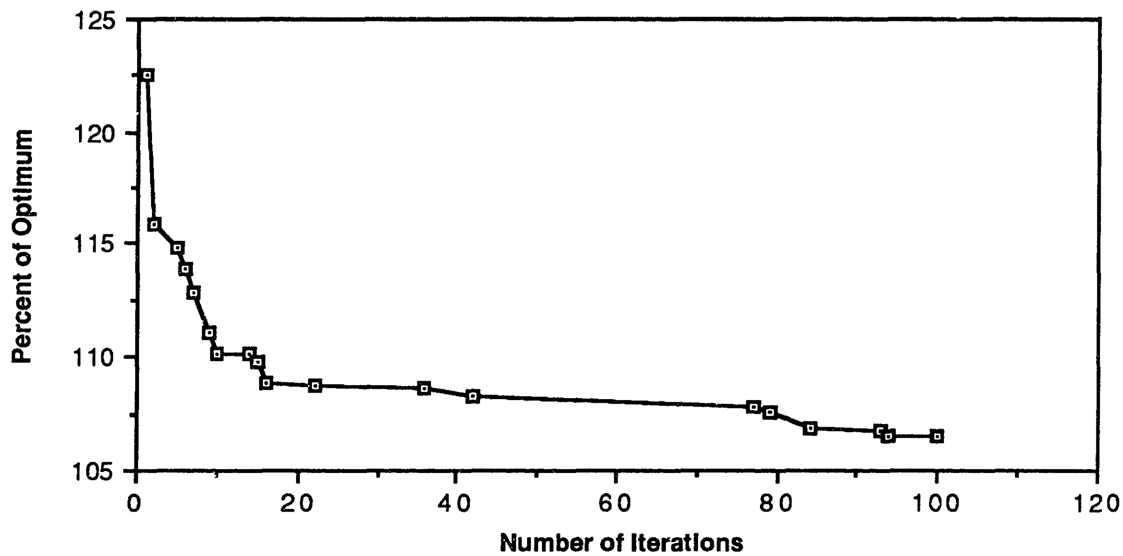
5.3.5 Producing Complete Solutions

Problems containing time constraints more complicated than the precedence constraints in the resource constrained multi-project scheduling problems may fail to produce complete solutions on the first iteration of an heuristic. As the iterative method searches, it will attempt to successfully schedule more and more jobs until a complete schedule is found. Test Problem 10, described in Appendix A, encounters this difficulty in finding a complete solution. Table A.35 and Figure A.26 show the average number of iterations each heuristic requires to find a complete solution. Averaged over all the heuristics, the randomized algorithm required significantly more iterations to find a complete schedule (13.6) than did the deterministic algorithm (6.4). This is in constrast to the fact that, after 100 iterations, the randomized heuristic usually produced better schedules than the deterministic algorithm (Table A.26 and Figure A.27).

5.3.6 Depth of Search

As the iterative algorithm progresses, better solutions appear less and less frequently. Figure 5.25 shows an example of how problem solution quality improves with the number of iterations. The data shown is from Problem 9. It is an average of the solution quality for 6 attempts of 100 iterations each, for the randomized version of Heuristic 6. It shows how average solution quality improves dramatically over the first 20 or so iterations, and then only improves slowly thereafter. (Of course, should the optimum be found, it will never find a better solution, no matter how long it continues iterating.)

Figure 5.25: Solution Improvement with Iterations



For Problems 7, 9, and 10, data was maintained to see whether a better solution would, on average, be obtained by performing 2 scheduling attempts of 50 iterations, and keeping the best solution, instead of 1 attempt of 100 iterations. Both options require the same amount of computational resources. As reported in Appendix A, for Problem 7, better results were obtained for all the heuristics (both deterministic and randomized versions) by stopping after 50 iterations. For Problems 9 and 10, however, results were mixed, usually favoring continuing to 100 iterations. This discrepancy can be explained because Problem 7 is much easier than Problems 9 and 10, and by the time 50 iterations have occurred, the algorithm is already at, or very near, optimum. Problems 9 and 10, however, are more difficult, and have a much large solution space than Problem 7. It is therefore likely that they will continue to find better solutions for a longer number of iterations.

5.3.7 Synthesis of Results

The results in Sections 5.3.1 to Section 5.3.6 point out several important facts: 1) that the deterministic versions of the heuristics strongly tend to produce superior initial solutions than do the randomized versions (Figure 5.19 and Figure 5.20); 2) that the quality of an initial solution produced by an algorithm is not indicative of the quality of the final solution it produces (Figure 5.24); 3) that the randomized versions of the heuristics strongly tend to produce superior final solutions than do the deterministic versions (Figure 5.22 and Figure 5.23); 4) that, for the final iterated deterministic solutions, none of the heuristic techniques (with the exceptions of Heuristic 9, **MCM**, and Heuristic 10, **CCM**, which utilize the compatibility matrix) produced significantly better solutions than did Heuristic 6, **JER**, which rates all jobs equally (Figure 5.22); and 5) that for problems in which it is difficult to find a complete schedule, the determinisitic versions of the iterated algorithms find complete solutions faster than do the randomized versions.

From these observations, several conclusions can be drawn:

1) That the power of the iterative technique comes from the technique itself, not the knowledge embedded in the heuristic (with the exceptions of Heuristics 9 and 10, as discussed in Section 5.3.8). This result can be explained by the realization that as the iterative technique progresses, the increase in priorities of the jobs (by the iterative search) "drowns out" the knowledge embedded in the heuristic, which has only a transient effect on solution quality during early iterations.

2) The superior performance of the initial solutions of the deterministic heuristics can probably be explained by the following analogy. Consider the problem of finding the average of two random integers: one chosen with equal probability from the integers between 0 and 39, and the other chosen with equal probability from the integers between 40 and 99. The expected value of the first integer will be 19.5, and the expected value of the second integer will be 69.5. The expected average value will therefore be 44.5, which is greater than 39.5, the point around which the integers were chosen with equal probability. This is because the interval above 39.5 is larger than the interval below 39.5. It should also be noted that the average value of 44.5 is still smaller than the midpoint of the total range of integers, namely 49.5.

A similar situation occurs with respect to the randomized initial solutions. If the deterministic initial solutions incorporate "knowledge" which makes them superior to a randomly chosen schedule, than the deterministic initial solution will lie below the midpoint of the range of solutions. The randomized version of the same heuristic will tend to find initial solutions centered about the deterministic solution (which is the "likeliest" randomized solution). Initial solutions produced by the randomized technique will not be uniformly distributed above and below the deterministic solution (rather, they will be clustered near it), but the conclusion drawn from the example still is valid: the average value of the randomized solution will be larger (i.e., worse) than the deterministic solution, given that the deterministic solution is better than a randomly generated solution (i.e., the randomized version of Heuristic 6, **JER**). Further, the average value of the initial randomized solution will be smaller (i.e., better) than a randomly generated initial solution if the initial deterministic solution is better than a randomly generated initial solution. Both of these conclusions are strongly supported by Figure 5.20.

3) The randomized versions of the iterative heuristics produce better final solutions because of their ability to "wander" from the direction indicated by the increased prioritization of the job priorities. While the deterministic versions tend to get stuck in "local minima," the random nature of the randomized heuristics allows them to "reach out" from these minima to find better solutions. This fact, (along with the fact that the deterministic algorithms have better initial solutions), also helps to explain why the deterministic algorithm tend to produce better early solutions, and tend to find complete solutions earlier, for problems in which finding a complete solution is difficult. During early iterations, before the algorithm has had a chance to find a local minimum, the priorities given to the jobs by the iterative algorithm are still small: the job weights assigned by the heuristics still contribute important knowledge. With the randomized heuristics, there is a probability that early iterative solutions (while the priorities are still small) will ignore both the knowledge initially embedded in the heuristic, as well as the knowledge gained through iteration. The very ability which enables the randomized versions of the heuristics to produce better solutions after a large number of iterations causes them to get sidetracked during early iterations and ignore knowledge which is very important.

5.3.8 Heuristics Utilizing the Compatibility Matrix

As illustrated in Figure 5.22, the two techniques which utilize the compatibility matrix (Heuristic 9, **MCM**, and Heuristic 10, **CCM**) find consistently better final (iterated) solutions than the deterministic versions of all the other heuristics, and they are also among the best of the randomized versions of the heuristics. The heuristics which utilize the compatibility matrix have a fundamentally different structure than the other heuristics. Whereas the iterated versions of the first eight heuristics effectively maintain data on the relative importance (priority) of ordering each job to be dispatched, the compatibility matrix maintains data on the relative importance for each pair of jobs to be dispatched one after the other. It is in this sense that the structure of Heuristics 9 and 10 are different.

A natural question to ask is whether Heuristics 9 and 10 work better because of this structure, or because the knowledge embedded in the compatibility matrix is significant. Considerable computation is necessary to generate the compatibility matrix (see Section 5.3.1), but if its power lies in its structure instead of in the information in the matrix, this computation could be eliminated. In order to determine this, a null model was prepared by replacing all the positive numbers in the compatibility matrix for Problems 9 and 10 by numbers randomly chosen from a uniform distribution between 1 and 100. Problems 9 and 10 were chosen because Heuristics 9 and 10 were extremely succesful for them. For each of these 2 problems, 7 such randomly generated matricies were prepared, and for each of these 7 matricies, 100 iterations of the deterministic version of Heuristic 9 were applied 3 times, and the results averaged. Table 5.7 shows the average results (expressed as percent of optimum) for each of the ten heuristics as well as the null model.

Table 5.7: Comparison of Null Model Results, Problems 9 and 10

Heuristic	Problem 9	Problem 10
1 SJF	107.609	109.964
2 LJF	106.972	104.076
3 MSLK	107.068	109.752
4 GRD	114.709	104.650
5 GRR	114.709	108.273
6 JER	107.402	108.560
7 LFT	108.182	108.937
8 EST	107.450	113.617
9 MCM	102.287	101.504
10 CCM	102.450	102.550
Average	107.884	107.188
Null Model	107.486	109.852

As can be seen from the above table, the null model performed significantly worse than the unmodified version of Heuristic 9. In fact, the null model performed comparably to Heuristic 6,

JER, which is, in fact, a null model for the other heuristics. The conclusion is therefore that Heuristics 9 and 10 perform better because of the data contained in the compatibility matrix (and the way in which the iterative algorithm makes use of this data).

5.3.9 Recommendations for Using the Iterative Algorithm

Based on the above results, the following guidelines are made for using the iterative method to find solutions to scheduling problems. A reasonable approach would be to use the iterated version of one or more heuristics. The number of iterations necessary depends on the particular problem, but on the order of 1/2 to 2 times the number of jobs being scheduled would seem to be a reasonable depth to search (note that the complexity of the entire algorithm is thus of the order of the fourth power of the number of jobs). Ideally, this technique would be applied over several (i.e., 3 - 5) trials to the same problem, and the best resulting solution would be used. The choice of whether to use deterministic or randomized iterations (or some combination), or of which heuristic to use, are not clearcut issues, but some recommendations are made below. It must be emphasized, however, that these decisions will have second order effects compared to the basic improvements which will be gained by using the iterative algorithm (in most any form).

As the deterministic version of an heuristic is better at finding good initial solutions, while the randomized version is better during later iterations, it is recommended that a hybrid approach be adopted, where during early iterations the heuristic is applied deterministically, and during latter iterations it is applied in the randomized fashion. This technique should combine the ability of the deterministic approach to use its knowledge to find good solutions quickly, while still keeping the ability of the randomized approach to get away from local minima during later iterations.

Finding the right point at which to switch from deterministic to randomized iterations is important, because if one switches too early, then there is a chance that one will not make full use of the "strengths" of the deterministic algorithm (see Section 5.3.7). On the other hand, if one switches too late, then the deterministic algorithm may have drifted too far in the wrong direction for the randomized algorithm to "wander out" and find better solutions. A good point for which to switch from deterministic to randomized iterations is after the deterministic algorithm stops producing significant improvement. For Problem 9 (which has 19 jobs), this typically occurs after about 15 to 20 iterations. Figure 5.25, although based on data for the randomized version of Heuristic 6, has a structure typical of how algorithm performance (either deterministic or randomized) will, on average, improve with the number of iterations. A graph such as shown in Figure 5.25, however, cannot be drawn for a particular problem until the problem has been solved many times and the results average. For use on a real problem, a method is needed to know where to switch before the problem has been solved.

Based on emperical testing, a reasonable approach would be to perform an initial trial where the algorithm switches from deterministic to randomized after a number of iterations equal to about 3/4 of the number of jobs. On subsequent trials, this switching point could be adjusted based on experience with previous trials. For example, if on the first trial one stopped finding significant improvement after a number of iterations equal to 1/2 the number of jobs, on the second trial one would switch to randomized iterations after that number of iterations.

The data generated from the test problems examined in this thesis clearly favors using the Maximum Compatibility Method or the Constrained Compatibility Method (Heuristics 9 and 10). However, for particular problem structures, it is always possible that other heuristics will produce superior results. Table 5.8 shows the expected improvement (shown as percent of optimum) which was achieved for 8 of the test problems using Heuristic 6, **JER**, and Heuristic 9, **MCM**. A parametric analysis to attempt to find which problem attributes correlate with the performance of different heuristics was beyond the scope of this study [see, Patterson, 1976]. However, from Table 5.8, it can be seen that Heuristic 9's advantage over Heuristic 6 is on Problems 9 and 10. These problems are characterized by having very few time constraints. Further, at any give time during scheduling, the total amount of resources demanded by all the pending jobs greatly exceeds the amout available. A quantity which tries to measure the overdemand for a resource is known as the Obstruction Factor, and is defined in Appendix A. It is on problems with large obstruction factors that the compatibility matrix method appears to performs strongest.

Table 5.8: Expected Solution Improvement with Iterative Algorithm

Problem	Average Randomly Chosen Initial Solution	Average After 100 Iterations of Heuristic 6 (Randomized)	Average After 100 Iterations of Heuristic 9 (Randomized)	Total Obstruction Factor
Prob 1	134.75	100	100	0.15
Prob 3	117.744	100	100	0.55
Prob 4	116.45	100	100	0.46
Prob 5	124.543	101.906	101.906	0.37
Prob 6	115.837	101.551	102.326	0.33
Prob 7	113.058	104.687	104.687	0.11
Prob 9	123.913	106.527	103.566	3.42
Prob 10	*	108.56	101.504	3.25

*Complete initial solutions were not possible for Problem 10.

One drawback to using Heuristics 9 and 10 is that generating the maximum compatibility matrix can require a substantial computational overhead. If necessary, another heuristic can be substituted. In any case, it might be prudent to try several heuristics, especially if a chosen heuristic does not appear to produce "good" results (although this can be difficult to access, *a priori*).

5.4 Application to a "Realistic" Scenario

The iterative algorithm was applied to a problems intended to simulate space station operations over a period of 48 hours. This problem involves requesting performance by 6 crewmembers of 93 jobs. The 6 crewmembers are divided into two groups of 3. The "Blue Team" includes Crewmembers 3, 4, and 6, and the "Gold Team" includes Crewmembers 1, 2, and 5. The two teams have independent (nonoverlapping) sleep periods.

Table 5.10A and Table 5.10B show the 93 jobs, the number of crewmembers which are required to perform them, and their minimum performance times, which are defined as the performance time of the crewmember (or crewmembers, if a job requires more than one) who can complete it fastest. Also shown in these tables are the amounts of four resources used by the jobs, and the audio/vibration status of each job. The four resources are labelled Power, Data Transmission, Computer Memory, and High Rate Multiplex. These are the same four resources which were used for Spacelab mission 1 [Grone and Mathis, 1980]. The total usage of each of the resources by the jobs is limited to 4000 units at any given time during the 48 hour flight plan. The Audio/Vibration constraint requires that no job which is noise or vibration sensitive (e.g. sleeping, or jobs requiring micro-gravity) be conducted at the same time as a noise or vibration generating job. Table 5.11 shows the performance time for each of the individual crewmembers on each of the jobs. A blank entry indicates that the crewmember is unrated (and hence unable) to perform the job. The jobs and their associated parameters presented in these tables are intended to illustrate a problem of similar complexity (within the limits of the scheduling software, as discussed in Section 6) to a real scheduling problem; however, no real Spacelab, Space Shuttle, or Space Station data was available, and hence the actual numbers used are entirely fictional.

Of the 93 jobs, 40 (Jobs 1 - 6 and Jobs 9 - 42) constitute "core" activities related to daily crewmember needs such as sleep, rest, hygiene and exercise periods, breakfast and dinner, and turnover briefings between shifts. With the exception of exercise periods, which can be scheduled independently for each crewmember, each of the core activities are performed together by all three crewmembers on the same team (the turnover briefings are performed together by all 6 crewmembers).

The core jobs are all linked together by a complex web of time constraints specifying earliest start times, latest start times, latest end times, precedence relations, and maximum and minimum

Table 5.10A: Resource Usages For Jobs 1 - 45

	JOB NAMES	PRIORITY	CREW REQ.	MIN. TIME	POWER USAGE	TRANS USAGE	MEMORY USAGE	MULT. USAGE	AUDIO LEVEL
1	*Gold Sleep Period 1	1	3	480					SENSITIVE
2	*Gold Sleep Period 2	1	3	480					SENSITIVE
3	*Blue Sleep Period 1	1	3	480					SENSITIVE
4	*Blue Sleep Period 2	1	3	480					SENSITIVE
5	*Gold Relax Period 1	1	3	30					NEUTRAL
6	*Gold Relax Period 2	1	3	30					NEUTRAL
7	*EVA Satellite Repair	1	2	465	1800	1205	1000	1585	NEUTRAL
8	*EVA Monitor/RMS Operator	1	1	360	600	1345	200	1300	NEUTRAL
9	*Blue Relax Period 1	1	3	30					NEUTRAL
10	*Blue Relax Period 2	1	3	30					NEUTRAL
11	*Gold Hygiene Period 1	1	3	30	150				GENERATING
12	*Gold Hygiene Period 2	1	3	30	150				GENERATING
13	*Blue Hygiene Period 1	1	3	30	150				NEUTRAL
14	*Blue Hygiene Period 2	1	3	30					GENERATING
15	*Gold Breakfast Period 1	1	3	30	300				NEUTRAL
16	*Gold Breakfast Period 2	1	3	30	300				NEUTRAL
17	*Blue Breakfast Period 1	1	3	30	300				NEUTRAL
18	*Blue Breakfast Period 2	1	3	30	300				NEUTRAL
19	*Gold Exercise Period 1a	1	1	90	1200				GENERATING
20	*Gold Exercise Period 1b	1	1	90	1200				GENERATING
21	*Gold Exercise Period 1c	1	1	90	1200				GENERATING
22	*Gold Exercise Period 2a	1	1	90	1200				GENERATING
23	*Gold Exercise Period 2b	1	1	90	1200				GENERATING
24	*Gold Exercise Period 2c	1	1	90	1200				GENERATING
25	*Blue Exercise Period 1a	1	1	90	1200				GENERATING
26	*Blue Exercise Period 1b	1	1	90	1200				GENERATING
27	*Blue Exercise Period 1c	1	1	90	1200				GENERATING
28	*Blue Exercise Period 2a	1	1	90	1200				GENERATING
29	*Blue Exercise Period 2b	1	1	90	1200				GENERATING
30	*Blue Exercise Period 2c	1	1	90	1200				GENERATING
31	*Gold Dinner Period 1	1	3	60	800				NEUTRAL
32	*Gold Dinner Period 2	1	3	60	800				NEUTRAL
33	*Blue Dinner Period 1	1	3	60	800				NEUTRAL
34	*Blue Dinner Period 2	1	3	60	800				NEUTRAL
35	*Gold Rest Period 1	1	3	30	350				NEUTRAL
36	*Gold Rest Period 2	1	3	30	350				NEUTRAL
37	*Blue Rest Period 1	1	3	30	350				NEUTRAL
38	*Blue Rest Period 2	1	3	30	350				NEUTRAL
39	*Turnover Briefing 1	1	6	10					NEUTRAL
40	*Turnover Briefing 2	1	6	10					NEUTRAL
41	*Turnover Briefing 3	1	6	10					NEUTRAL
42	*Turnover Briefing 4	1	6	10					NEUTRAL
43	*Vacuum Air Filter 1	1	1	30	1500				GENERATING
44	*Vacuum Air Filter 2	1	1	30	1500				GENERATING
45	*Ground Briefing	1	2	30	1000	3000			NEUTRAL

Table 5.10B: Resource Usages For Jobs 46 - 93

	JOB NAMES	PRIORITY	CREW REQ.	MIN. TIME	POWER USAGE	TRANS USAGE	MEMORY USAGE	MULT. USAGE	AUDIO LEVEL
46	*PAO Video Downlink	1	2	45	1000	3000			NEUTRAL
47	*Systems Checkout	1	2	60	1500	1000	2000	1000	NEUTRAL
48	*Cancel Postage Stamps	1	1	35					NEUTRAL
49	*Clean Animal Compartments	1	1	180	1200	1000	400	600	NEUTRAL
50	*Material Processing, Step 1	1	2	40	1000	1000	500		NEUTRAL
51	*Material Processing, Step 2a	1	2	120	1500	1500	670	1000	SENSITIVE
52	*Material Processing, Step 2b	1	2	120	1500	1500	670	1000	SENSITIVE
53	*Materials Processing, Step 2c	1	2	120	1500	1500	670	1000	SENSITIVE
54	*Materials Processing, Step 3a	1	0	300	2000			1000	SENSITIVE
55	*Material Processing, Step 3b	1	0	300	2000			1000	SENSITIVE
56	*Materials Processing, Step 3c	1	0	300	2000			1000	SENSITIVE
57	*Materials Processing, Step 4a	1	1	30	500	500			NEUTRAL
58	*Materials Processing, Step 4b	1	1	30	500	500			NEUTRAL
59	*Materials Processing, Step 4c	1	2	30	500	500			NEUTRAL
60	*Materials Processing, Step 5	1	2	60	1700				NEUTRAL
61	*Earth Observation A	1	2	15	1500	1980	2500	1300	NEUTRAL
62	*Earth Observation B	1	2	15	1500	1980	2500	1300	NEUTRAL
63	*Earth Observation C	1	2	15	1500	1980	2500	1300	NEUTRAL
64	Bulk Crystal	1	2	195	922	2601	2980	1340	GENERATING
65	Alloy Solidification	1	5	80	6	3991	2542	493	GENERATING
66	*Autoignition Furnace	1	3	19	378	107	1732	1473	GENERATING
67	*Bioreactor/Incubator	1	3	55	89	694	668	677	NEUTRAL
68	*Acoustic Levitator	1	3	120	789	1794	134	498	SENSITIVE
69	*Atmospheric Microphysics	1	1	23	171	466	56	998	SENSITIVE
70	*Bridgman, Large	1	4	40	317	3050	1179	1081	GENERATING
71	Bridgman, Small	1	4	44	659	361	1035	47	SENSITIVE
72	*Continuous Flow Electrophoresis	1	1	102	795	3037	350	303	NEUTRAL
73	*Critical Point Phenomena	1	1	161	102	2957	1636	364	SENSITIVE
74	*Droplet/Spray Burning	1	2	70	998	2113	2917	1468	NEUTRAL
75	*Electroepitaxy	1	2	100	505	411	2044	378	GENERATING
76	Electrostatic Levitator	1	4	228	77	285	1971	1021	SENSITIVE
77	*EM Levitator	1	1	108	480	1209	1354	1451	GENERATING
78	*Float Zone	1	2	19	279	660	2520	658	NEUTRAL
79	Fluid Physics	1	4	165	513	2481	1684	1660	NEUTRAL
80	*Free Float	1	2	135	24	3637	260	1908	NEUTRAL
81	*High Temperature Physics	1	4	24	532	965	2898	1252	GENERATING
82	Isoelectric Focusing	1	1	189	285	911	598	784	GENERATING
83	*Latex Reactor	1	2	55	600	230	200	900	NEUTRAL
84	Membrane Production	1	5	120	100	600	300	400	GENERATING
85	*Optical Fiber Pulling	1	1	140	670	207	239	921	SENSITIVE
86	*Organic and Polymer Crystal Growth	1	3	110	237	888	279	109	GENERATING
87	*Premixed Gas Combustion	1	1	80	440	100	900	289	NEUTRAL
88	*Protein Crystal Growth	1	2	79	870	667	200	300	SENSITIVE
89	*Rotating Spherical Convection	1	3	23	567	900	200	276	NEUTRAL
90	Solid Furnace Burning	1	4	89	456	89		111	GENERATING
91	*Solution Crystal	1	2	75	230	300	650	400	SENSITIVE
92	*Vapor Crystal	1	3	25	400	200	500	200	NEUTRAL
93	*Variable Flow Shell Generator	1	2	95	300	700	600	200	NEUTRAL

Table 5.11: Performance Times of the Crewmembers

	JOBS	CREW1	CREW2	CREW3	CREW4	CREW5	CREW6
1	Gold Sleep Period 1	480	480			480	
2	Gold Sleep Period 2	480	480			480	
3	Blue Sleep Period 1			480	480		480
4	Blue Sleep Period 2			480	480		480
5	Gold Relax Period 1	30	30			30	
6	Gold Relax Period 2	30	30			30	
7	EVA Satellite Repair		465	465	465		
8	EVA Monitor/RMS Operator		360	360	360	360	360
9	Blue Relax Period 1			30	30		30
10	Blue Relax Period 2			30	30		30
11	Gold Hygiene Period 1	30	30			30	
12	Gold Hygiene Period 2	30	30			30	
13	Blue Hygiene Period 1			30	30		30
14	Blue Hygiene Period 2			30	30		30
15	Gold Breakfast Period 1	30	30			30	
16	Gold Breakfast Period 2	30	30			30	
17	Blue Breakfast Period 1			30	30		30
18	Blue Breakfast Period 2			30	30		30
19	Gold Exercise Period 1a					90	
20	Gold Exercise Period 1b	90					
21	Gold Exercise Period 1c		90				
22	Gold Exercise Period 2a					90	
23	Gold Exercise Period 2b	90					
24	Gold Exercise Period 2c		90				
25	Blue Exercise Period 1a						90
26	Blue Exercise Period 1b				90		
27	Blue Exercise Period 1c			90			
28	Blue Exercise Period 2a						90
29	Blue Exercise Period 2b				90		
30	Blue Exercise Period 2c			90			
31	Gold Dinner Period 1	60	60			60	
32	Gold Dinner Period 2	60	60			60	
33	Blue Dinner Period 1			60	60		60
34	Blue Dinner Period 2			60	60		60
35	Gold Rest Period 1	30	30			30	
36	Gold Rest Period 2	30	30			30	
37	Blue Rest Period 1			30	30		30
38	Blue Rest Period 2			30	30		30
39	Turnover Briefing 1	10	10	10	10	10	10
40	Turnover Briefing 2	10	10	10	10	10	10
41	Turnover Briefing 3	10	10	10	10	10	10
42	Turnover Briefing 4	10	10	10	10	10	10
43	Vacuum Air Filter 1	30	30	30	30	30	30
44	Vacuum Air Filter 2	30	30	30	30	30	30
45	Ground Briefing		30	30	30	30	30
46	PAO Video Downlink	45	45	45	45	45	45
47	Systems Checkout		60	60	60	60	60
48	Cancel Postage Stamps	35	35	35	35	35	35
49	Clean Animal Compartments		180	180	180		
50	Material Processing, Step 1	40	40	40	40		
51	Material Processing, Step 2a	110	120	120	120		
52	Material Processing, Step 2b	110	120	120	120		
53	Materials Processing, Step 2c	110	120	120	120		
54	Materials Processing, Step 3a	300	300	300	300	300	300
55	Material Processing, Step 3b	300	300	300	300	300	300
56	Materials Processing, Step 3c	300	300	300	300	300	300
57	Materials Processing, Step 4a	30	30	30	30		
58	Materials Processing, Step 4b	30	30	30	30		
59	Materials Processing, Step 4c	30	30	30	30		
60	Materials Processing, Step 5	60	60	60	60		
61	Earth Observation A	15	15	15	15		
62	Earth Observation B	15	15	15	15		
63	Earth Observation C	15	15	15	15		
64	Bulk Crystal	195	200	190	196		230
65	Alloy Solidification	75	75	65	75	80	85
66	Autoignition Furnace	20	25	19	19	19	19
67	Bioreactor/Incubator	42	45	55	69	69	69
68	Acoustic Levitator	120	95	85	120	120	
69	Atmospheric Microphysics	23		30	40	29	30
70	Bridgman, Large	40	40	40	40	40	40
71	Bridgman, Small	44	40	44	44	40	44
72	Continuous Flow Electrophoresis	102	102	102	102	102	102
73	Critical Point Phenomena	161	161	161	161	161	161
74	Droplet/Spray Burning	70	70	70	70	70	70
75	Electroepitaxy	100	90	100	100	100	100
76	Electrostatic Levitator	228	228	228	228	200	228
77	EM Levitator	108	108	120	108	108	125
78	Float Zone	19	35	19	19	18	19
79	Fluid Physics	165	165	165	165	165	165
80	Free Float	144	135	145	144	130	144
81	High Temperature Physics	24	40	30	24	24	24
82	Isoelectric Focusing	216	189	216	216	216	216
83	Latex Reactor	55	55	55	55	55	55
84	Membrane Production	120	120	120	120	120	120
85	Optical Fiber Pulling	140	140	140	140	140	140
86	Organic and Polymer Crystal Growth	110	110	110	110	110	110
87	Premixed Gas Combustion	80	80	80	80	80	80
88	Protein Crystal Growth	79	79	79	79	79	79
89	Rotating Spherical Convection	23	23	23	23	23	23
90	Solid Furnace Burning	89	89	89	89	89	89
91	Solution Crystal	75	75	75	75	75	75
92	Vapor Crystal	25	25	25	25	25	25
93	Variable Flow Shell Generator	95	95	95	95	95	95

intervals between the starts of jobs. For example, it is always required that exercise periods be held at least 1 hour and 30 minutes after breakfast and at least 2 hours and 30 minutes before dinner. Effort was made to put reasonable conditions into the schedule while still allowing for flexibility. Instead of rigidly fixing the sleep periods of the crewmembers, their initial sleep period was specified to start within a window of 3 1/2 hours, and their subsequent sleep periods were set to begin no less than 23 hours nor more than 25 hours after the start of their first sleep period. This approximates a normal work/rest cycle while still maintaining the flexibility necessary to move sleep periods to accommodate the meeting of mission goals.

The time constraints pertaining to the core activities are shown in Figure 5.26. Each of the core activities is represented by a node on this graph. When jobs are constrained by earliest start time (EST), latest start time (LST) or latest end time (LET) constraints, the nodes corresponding to these jobs are so labelled. Precedence constraints between jobs are shown as unlabelled arcs which have an arrow at the node of the "following" job. For example, towards the center of Figure 5.26 one can see that Job 2 (Gold Team Sleep Period 2) cannot start earlier than 1470 minutes (24 1/2 hours) after the start of the timeline. It can also be seen that Job 2 must be preceded by Job 6 (Gold Team Relax Period 2) and Job 40 (Turnover Briefing 2), and that Job 41 (Turnover Briefing 3) and Job 12 (Gold Team Hygiene Period 2) must follow Job 2. Constraints specifying maximum and minimum time intervals between jobs are indicated by labelled arcs which have an arrow in the middle of the arc. For example, it is seen that Job 1 (Gold Team Sleep Period 1) must start at a minimum of 1380 minutes (23 hours) and a maximum of 1500 minutes (25 hours) before Job 2. It can also be seen that Job 4 (Blue Team Sleep Period 2) must start at least 540 minutes (9 hours) after the start of Job 2. The complexity of the time constraints in Figure 5.26 is clearly much greater than those in the problems taken from the literature which are detailed in Appendix A and which were analyzed in Section 5.3.

Figure 5.27 shows time constraint networks for 4 other groups of jobs. Jobs 43 and 44 are jobs which involve the vacuuming of an air filter, and time constraints specify that this be done once within the first 24 hour period and once within the second 24 hour period. Constraints also require that these two performances be separated by at least 12 hours and by at most 36 hours.

Job 7 (EVA Satellite Repair) and Job 8 (EVA Monitor/RMS Operator) represent the scheduling of a two crewmember extravehicular activity, with a third crewmember providing operational support. Job 7 includes 45 minutes for EVA preparation, a 6 hour EVA, and 1 hour

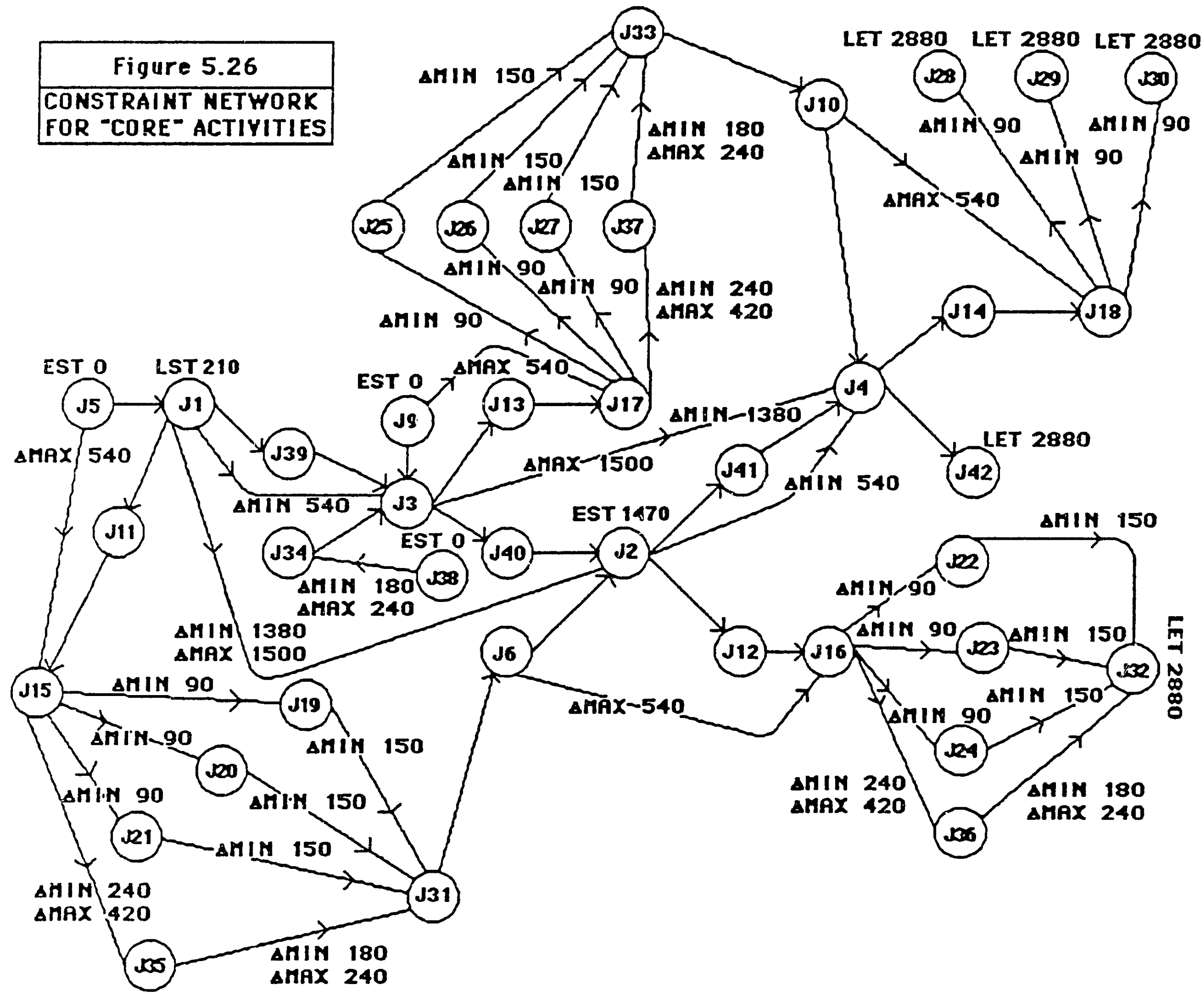
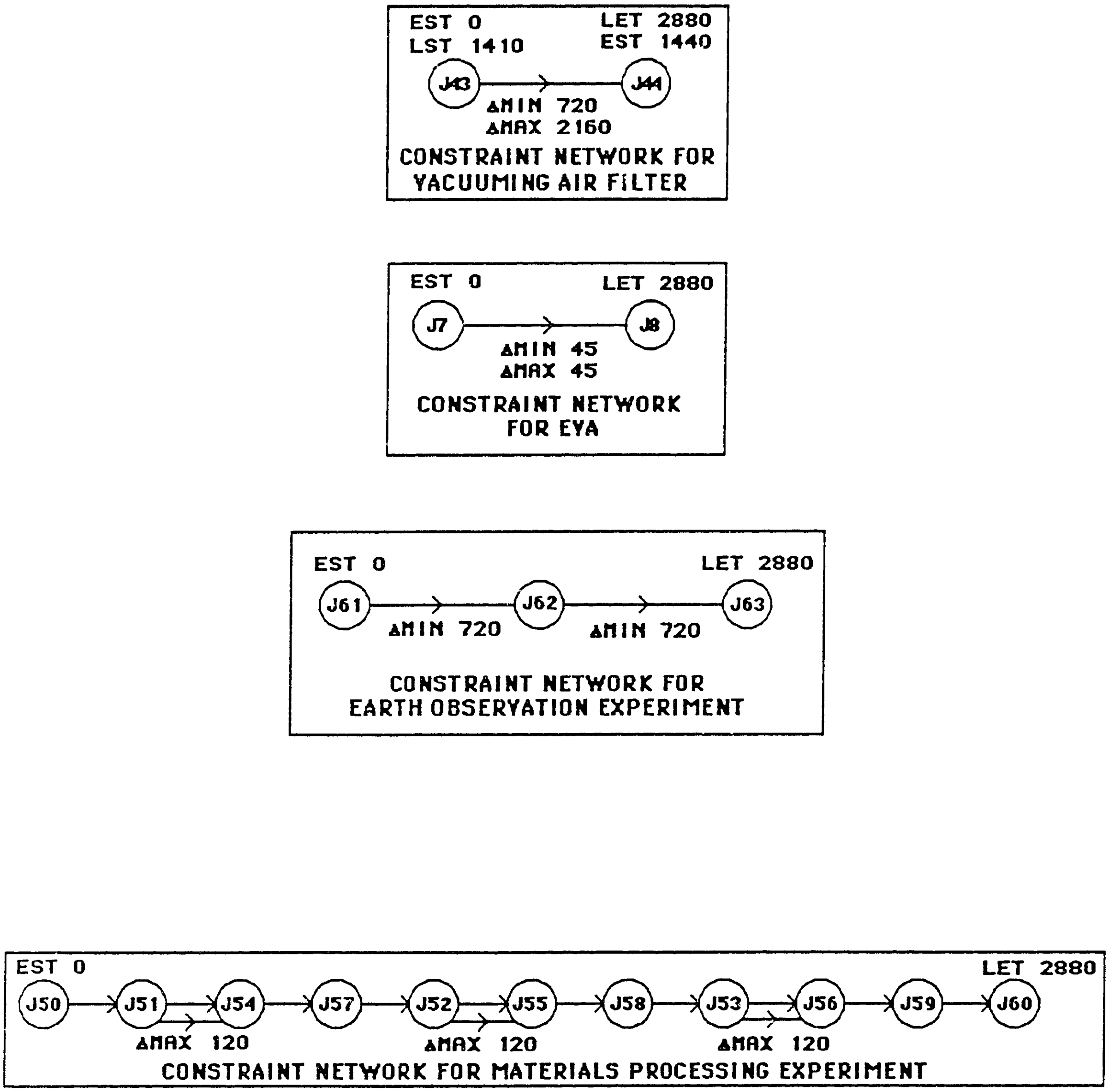


Figure 5.26
CONSTRAINT NETWORK FOR "CORE" ACTIVITIES

Figure 5.27: Constraint Networks for Space Station Planning Problem



for post EVA suit removal and stowage. Job 8 is thus constrained to correspond with the 6 hours of actual EVA operation.

Jobs 61, 62, and 63 correspond to parts of an earth observation experiment, with each portion of the second and third parts of the experiment starting at least 12 hours after the previous part.

Jobs 50 through 60 correspond to a materials processing experiment. Job 50 is a step which sets up the experiment, and is followed by three repetitions (performances) of the experiment, with each repetition composed of three steps. The first step is a sample preparation (Job 51 for the first performance, Job 52 for the second, and Job 53 for the third), which is constrained to be immediately followed by a step in which the sample is processed (for example, by baking it in a furnace) (Jobs 54, 55, and 56). This step does not require any crewmembers, but nonetheless consumes resources. The third step in each performance of the experiment is to remove and analyze the sample (Jobs 57, 58, and 59). Job 60 is for cleanup, disassembly, and stowage of the experimental apparatus.

For the first 63 jobs (which include all the core activities, the EVA, all the experiments discussed above, and 5 additional activities) a pre-scheduling analysis was performed to assure that a schedule would exist which could successfully schedule all 63 jobs, meeting all resource and time constraints. To these 63 jobs, 30 additional experiments were added, but no effort was made to assure that it would be possible to successfully incorporate these new jobs into the 48 hour schedule.

The iterative algorithm was tested first on the initial 63 job problem, and then on the entire 93 jobs. For the 63 job problem, the scheduler (see Section 6.2) was requested to perform 3 trials of 126 iterations each. For Heuristic 3, the Minimum Slack Method, on each of the three trials all 63 jobs were successfully scheduled. Among the three trials, the largest number of iterations which were required to find a complete scheduling was 19. The best complete schedule found had the last job completed by 46 hours and 40 minutes after the start of the scheduling period.

Heuristic 5, the Greatest Remaining Resource Requirement Heuristic, found a complete schedule on only 1 of the three trials (after 16 iterations). On the other two trials, only 41 and 45 jobs were scheduled, respectively, after 126 iterations. Heuristic 6, the Jobs Equally Rated Heuristic, had successfully scheduled only 43 jobs after each of the trials, and Heuristic 9, the

Maximum Compatibility Method, was only successful at scheduling 42 of the jobs after each of the three trials. Finally, Heuristic 10, the Constrained Compatibility Method, successfully scheduled all 63 jobs on two out of the three trials (after 97 and 122 iterations, respectively).

Based upon these results, Heuristics 3 and 10 were chosen to examine the full 93 job problem. Unexpectedly, both of these heuristics initially performed no better on the 93 job problem than on the 63 job problem. Heuristic 3 found solutions incorporating at most 59 jobs, while the best solution found by Heuristic 10 (utilizing a hybrid deterministic/randomized algorithm, as discussed in Section 5.3.9) incorporated 65 jobs.

Examination of the problem data revealed the reason why the algorithm was having difficulty finding superior solutions. Eight of the jobs (Jobs 64, 65, 71, 76, 79, 82, 84, and 90) were either difficult or impossible to feasibly incorporate in with the initial 63 jobs. If any of these jobs were encountered by the iterative algorithm, it would attempt to "work these jobs into the schedule," ignoring other jobs which it had not yet even attempted to schedule.

Two methods were used circumvent this problem. The first method was to take the schedule found with the iterative search, and then attempt to schedule in each of the remaining jobs, skipping any jobs which could not be feasibly incorporated into the schedule. This method was successful in scheduling 84 of the 93 jobs.

The second method also involved using the schedule found with the iterative search as a "base" upon which to add the other jobs. However, instead of making a single attempt to schedule in the final 30 jobs, the priorities of the 8 jobs causing difficulties were reduced , and then the iterative algorithm was used to attempt to find an efficient scheduling of the 30 unscheduled jobs. By manually reducing the priorities of the 8 jobs, the iterative algorithm would not consider them until it had successfully scheduled in the other 22 jobs. This is what did, in fact, happen, with a final schedule being produced incorporating 85 of the 93 jobs. Table 5.12A and Table 5.12B show, for each of the jobs, the order in which the job was dispatched in this final schedule, the start time and end time of each job, and the crewmember(s) to which each job was assigned.

Table 5.13 shows the final schedule in numerical form. A timeline in the rightmost column shows time units at each point in which there is a change in crewmember assignment or in resource use level. The first six columns show which job each of the crewmembers is doing at

Table 5.12A: Time and Crew Assignment for Jobs 1 - 45

	JOBS	ORDER	START	END	CREW1	CREW2	CREW3	CREW4	CREW5	CREW6
1	Gold Sleep Period 1	2	30	510	X	X			X	
2	Gold Sleep Period 2	26	1470	1950	X	X			X	
3	Blue Sleep Period 1	12	610	1090			X	X		X
4	Blue Sleep Period 2	38	2050	2530			X	X		X
5	Gold Relax Period 1	1	0	30	X	X			X	
6	Gold Relax Period 2	25	1440	1470	X	X			X	
7	EVA Satellite Repair	43	1510	1975			X	X		
8	EVA Monitor/RMS Operator	44	1555	1915						X
9	Blue Relax Period 1	11	580	610			X	X		X
10	Blue Relax Period 2	37	2020	2050			X	X		X
11	Gold Hygiene Period 1	3	510	540	X	X			X	
12	Gold Hygiene Period 2	30	1950	1980	X	X			X	
13	Blue Hygiene Period 1	13	1090	1120			X	X		X
14	Blue Hygiene Period 2	51	2530	2560			X	X		X
15	Gold Breakfast Period 1	4	540	570	X	X			X	
16	Gold Breakfast Period 2	32	1980	2010	X	X			X	
17	Blue Breakfast Period 1	14	1120	1150			X	X		X
18	Blue Breakfast Period 2	52	2560	2590			X	X		X
19	Gold Exercise Period 1a	16	1090	1180					X	
20	Gold Exercise Period 1b	17	1090	1180	X					
21	Gold Exercise Period 1c	15	1090	1180		X				
22	Gold Exercise Period 2a	39	2530	2620					X	
23	Gold Exercise Period 2b	40	2530	2620	X					
24	Gold Exercise Period 2c	41	2530	2620		X				
25	Blue Exercise Period 1a	22	1210	1300						X
26	Blue Exercise Period 1b	27	1300	1390				X		
27	Blue Exercise Period 1c	23	1210	1300			X			
28	Blue Exercise Period 2a	54	2650	2740						X
29	Blue Exercise Period 2b	55	2650	2740				X		
30	Blue Exercise Period 2c	56	2740	2830			X			
31	Gold Dinner Period 1	20	1240	1300	X	X			X	
32	Gold Dinner Period 2	57	2800	2860	X	X			X	
33	Blue Dinner Period 1	36	1450	1510			X	X		X
34	Blue Dinner Period 2	10	190	250			X	X		X
35	Gold Rest Period 1	19	580	610	X	X			X	
36	Gold Rest Period 2	49	2140	2170	X	X			X	
37	Blue Rest Period 1	35	1150	1180			X	X		X
38	Blue Rest Period 2	6	40	70			X	X		X
39	Turnover Briefing 1	7	570	580	X	X	X	X	X	X
40	Turnover Briefing 2	18	1180	1190	X	X	X	X	X	X
41	Turnover Briefing 3	33	2010	2020	X	X	X	X	X	X
42	Turnover Briefing 4	58	2860	2870	X	X	X	X	X	X
43	Vacuum Air Filter 1	21	0	30						X
44	Vacuum Air Filter 2	42	1950	1980						X
45	Ground Briefing	63	370	400				X		X

Table 5.12B: Time and Crew Assignment for Jobs 46 - 93

	JOBS	ORDER	START	END	CREW1	CREW2	CREW3	CREW4	CREW5	CREW6
46	PAO Video Downlink	61	325	370			X			X
47	Systems Checkout	60	265	325				X		X
48	Cancel Postage Stamps	62	70	105						X
49	Clean Animal Compartments	53	2620	2800		X				
50	Material Processing, Step 1	5	0	40			X	X		
51	Material Processing, Step 2a	8	70	190			X	X		
52	Material Processing, Step 2b	29	610	730	X	X				
53	Materials Processing, Step 2c	45	2020	2140	X	X				
54	Materials Processing, Step 3a	9	190	490						
55	Material Processing, Step 3b	31	730	1030						
56	Materials Processing, Step 3c	46	2140	2440						
57	Materials Processing, Step 4a	28	490	520			X			
58	Materials Processing, Step 4b	34	1030	1060	X					
59	Materials Processing, Step 4c	47	2440	2470	X	X				
60	Materials Processing, Step 5	48	2470	2530	X	X				
61	Earth Observation A	24	250	265			X	X		
62	Earth Observation B	50	730	745	X	X				
63	Earth Observation C	59	1975	1990			X	X		
64	Bulk Crystal									
65	Alloy Solidification									
66	Autoignition Furnace	84	1990	2009			X	X		X
67	Bioreactor/Incubator	66	745	814	X	X			X	
68	Acoustic Levitator	73	2305	2425	X	X			X	
69	Atmospheric Microphysics	82	265	295			X			
70	Bridgman, Large	69	1190	1230	X	X		X	X	
71	Bridgman, Small									
72	Continuous Flow Electrophoresis	76	2425	2527					X	
73	Critical Point Phenomena	70	865	1026		X				
74	Droplet/Spray Burning	65	400	470			X			X
75	Electroepitaxy	64	1300	1400	X				X	
76	Electrostatic Levitator									
77	EM Levitator	75	2740	2848				X		
78	Float Zone	85	1190	1209			X			X
79	Fluid Physics									
80	Free Float	72	2170	2305		X			X	
81	High Temperature Physics	81	2620	2644	X			X	X	X
82	Isoelectric Focusing									
83	Latex Reactor	79	2740	2795	X				X	
84	Membrane Production									
85	Optical Fiber Pulling	71	944	1084					X	
86	Organic and Polymer Crystal Growth	74	1300	1410		X	X			X
87	Premixed Gas Combustion	78	105	185						X
88	Protein Crystal Growth	68	865	944	X				X	
89	Rotating Spherical Convection	83	1410	1433		X	X			X
90	Solid Furnace Burning									
91	Solution Crystal	67	790	865	X	X				
92	Vapor Crystal	80	1400	1425	X			X	X	
93	Variable Flow Shell Generator	77	470	565				X		X

Table 5.13: Schedule Incorporating 85 Jobs

CREW1	CREW2	CREW3	CREW4	CREW5	CREW6	POWER	TRANS	MEMORY	MULT	AUDIO	TIME
5	5	50	50	5	43	2500	1000	500	0	1	0
1	1	50	50	1	0	1000	1000	500	0	-1	30
1	1	38	38	1	38	350	0	0	0	-1	40
1	1	51	51	1	48	1500	1500	670	1000	-1	70
1	1	51	51	1	87	1940	1600	1570	1289	-1	105
1	1	51	51	1	0	1500	1500	670	1000	-1	185
1	1	34	34	1	34	2800	0	0	1000	-1	190
1	1	61	61	1	0	3500	1980	2500	2300	-1	250
1	1	69	47	1	47	3671	1466	2056	2998	-1	265
1	1	0	47	1	47	3500	1000	2000	2000	-1	295
1	1	46	0	1	46	3000	3000	0	1000	-1	325
1	1	0	45	1	45	3000	3000	0	1000	-1	370
1	1	74	0	1	74	2998	2113	2917	2468	-1	400
1	1	0	93	1	93	2300	700	600	1200	-1	470
1	1	57	93	1	93	800	1200	600	200	-1	490
1	1	57	93	1	93	800	1200	600	200	-1	495
11	11	57	93	11	93	950	1200	600	200	1	510
11	11	0	93	11	93	450	700	600	200	1	520
15	15	0	93	15	93	600	700	600	200	0	540
15	15	0	0	15	0	300	0	0	0	0	565
39	39	39	39	39	39	0	0	0	0	0	570
35	35	9	9	35	9	350	0	0	0	0	580
52	52	3	3	0	3	1500	1500	670	1000	-1	610
0	52	3	3	0	3	1500	1500	670	1000	-1	720
62	62	3	3	0	3	3500	1980	2500	2300	-1	730
67	67	3	3	67	3	2089	694	668	1677	-1	745
0	67	3	3	67	3	2089	694	668	1677	-1	787
91	91	3	3	67	3	2319	994	1318	2077	-1	790
91	91	3	3	0	3	2230	300	650	1400	-1	814
88	73	3	3	88	3	2972	3624	1836	1664	-1	865
0	73	3	3	85	3	2772	3164	1875	2285	-1	944
0	0	3	3	85	3	2670	207	239	1921	-1	1026
58	0	3	3	85	3	1170	707	239	921	-1	1030
0	0	3	3	85	3	670	207	239	921	-1	1060
0	0	3	3	0	3	0	0	0	0	-1	1084
20	21	13	13	19	13	3750	0	0	0	1	1090
20	21	17	17	19	17	3900	0	0	0	1	1120
20	21	37	37	19	37	3950	0	0	0	1	1150
40	40	40	40	40	40	0	0	0	0	0	1180
70	70	78	70	70	78	596	3710	3699	1739	1	1190
70	70	0	70	70	0	317	3050	1179	1081	1	1209
70	70	27	70	70	25	2717	3050	1179	1081	1	1210
0	0	27	0	0	25	2400	0	0	0	1	1230
31	31	27	0	31	25	3200	0	0	0	1	1240
75	86	86	26	75	86	1942	1299	2323	487	1	1300
75	86	86	0	75	86	742	1299	2323	487	1	1390
92	86	86	92	92	86	637	1088	779	309	1	1400
92	89	89	92	92	89	967	1100	700	476	0	1410
0	89	89	0	0	89	567	900	200	276	0	1425
0	0	0	0	0	0	0	0	0	0	0	1433
6	6	0	0	6	0	0	0	0	0	0	1440
6	6	33	33	6	33	800	0	0	0	0	1450
2	2	33	33	2	33	800	0	0	0	-1	1470
2	2	7	7	2	0	1800	1205	1000	1585	-1	1510
2	2	7	7	2	8	2400	2550	1200	2885	-1	1555
2	2	7	7	2	0	1800	1205	1000	1585	-1	1915
12	12	7	7	12	44	3450	1205	1000	1585	1	1950
12	12	63	63	12	44	3150	1980	2500	1300	1	1975
16	16	63	63	16	0	1800	1980	2500	1300	0	1980
16	16	66	66	16	66	678	107	1732	1473	1	1990
16	16	0	0	16	0	300	0	0	0	0	2009
41	41	41	41	41	41	0	0	0	0	0	2010
53	53	10	10	0	10	1500	1500	670	1000	-1	2020
53	53	4	4	0	4	1500	1500	670	1000	-1	2050
0	53	4	4	0	4	1500	1500	670	1000	-1	2130
36	36	4	4	36	4	2350	0	0	1000	-1	2140
0	80	4	4	80	4	2024	3637	260	2908	-1	2170
0	80	4	4	0	4	2024	3637	260	2908	-1	2300
68	68	4	4	68	4	2789	1794	134	1498	-1	2305
68	0	4	4	68	4	2789	1794	134	1498	-1	2400
0	0	4	4	72	4	2795	3037	350	1303	-1	2425
59	59	4	4	72	4	1295	3537	350	303	-1	2440
60	60	4	4	72	4	2495	3037	350	303	-1	2470
60	60	4	4	0	4	1700	0	0	0	-1	2527
23	24	14	14	22	14	3600	0	0	0	1	2530
23	24	18	18	22	18	3900	0	0	0	1	2560
23	24	0	0	22	0	3600	0	0	0	1	2590
81	49	0	81	81	81	1732	1965	3298	1852	1	2620
0	49	0	0	0	0	1200	1000	400	600	0	2644
0	49	0	29	0	28	3600	1000	400	600	1	2650
83	49	30	77	83	0	3480	2439	1954	2951	1	2740
0	49	30	77	0	0	2880	2209	1754	2051	1	2795
32	32	30	77	32	0	2480	1209	1354	1451	1	2800
32	32	0	77	32	0	1280	1209	1354	1451	1	2830
32	32	0	0	32	0	800	0	0	0	0	2848
42	42	42	42	42	42	0	0	0	0	0	2860
0	0	0	0	0	0	0	0	0	0	0	2870
0	0	0	0	0	0	0	0	0	0	0	2880

the corresponding time in the timeline, until the next time listed in the timeline (a 0 corresponds to no job at that time). The next four columns in the table indicate resource usage at the times given, and the eleventh column indicates a -1 if a noise/vibration sensitive activity is occurring, a 1 if a noise/vibration generating activity is occurring, and a 0 otherwise. For example, it can be seen that from time 814 to time 865 Crewmembers 1 and 2 are performing Job 91, Crewmembers 3, 4, and 6 are performing Job 3, and Crewmember 5 is unoccupied. Also at this time, the resource levels are 2230, 300, 650, and 1400. The -1 in the tenth column indicates that at least one of the ongoing jobs is noise sensitive. Table 5.14 shows the amount of time each crewmember is occupied during the 48 hour mission. For each crewmember, a total of 25 hours and 40 minutes of this time is taken up by the core activities.

Table 5.14: Total Crewmember Workloads

Crewmember	Workload
C1 Russ Howard	40 hours, 55 minutes
C2 Clifford Kurtzman	45 hours, 49 minutes
C3 David Akin	42 hours, 21 minutes
C4 John Spofford	43 hours, 16 minutes
C5 Edith Erlanson	40 hours, 24 minutes
C6 Mary Bowden	42 hours, 50 minutes
Average:	42 hours, 35.8 minutes

Figures 5.28A throught 5.28H graphically show this schedule. Each figure shows a successive 6 hour interval in the 48 hour timeline. Each crewmember (**C1** through **C6**) has a graphical timeline which shows the jobs to which he or she is assigned during the interval encompassed by the figure. Also shown is a graphical illustration of resource usage throughout the interval. The line denoted AC (for audio constraint) displays whether or not any of the jobs which are scheduled at any particular time are noise or vibration generating (striped) or sensitive (gray). The timeline denote **C0** displays the scheduling of jobs which do not require any crewmembers. Section 6.2.1 provides a fuller interpretation of these figures as well as a description of the software used to generate them.

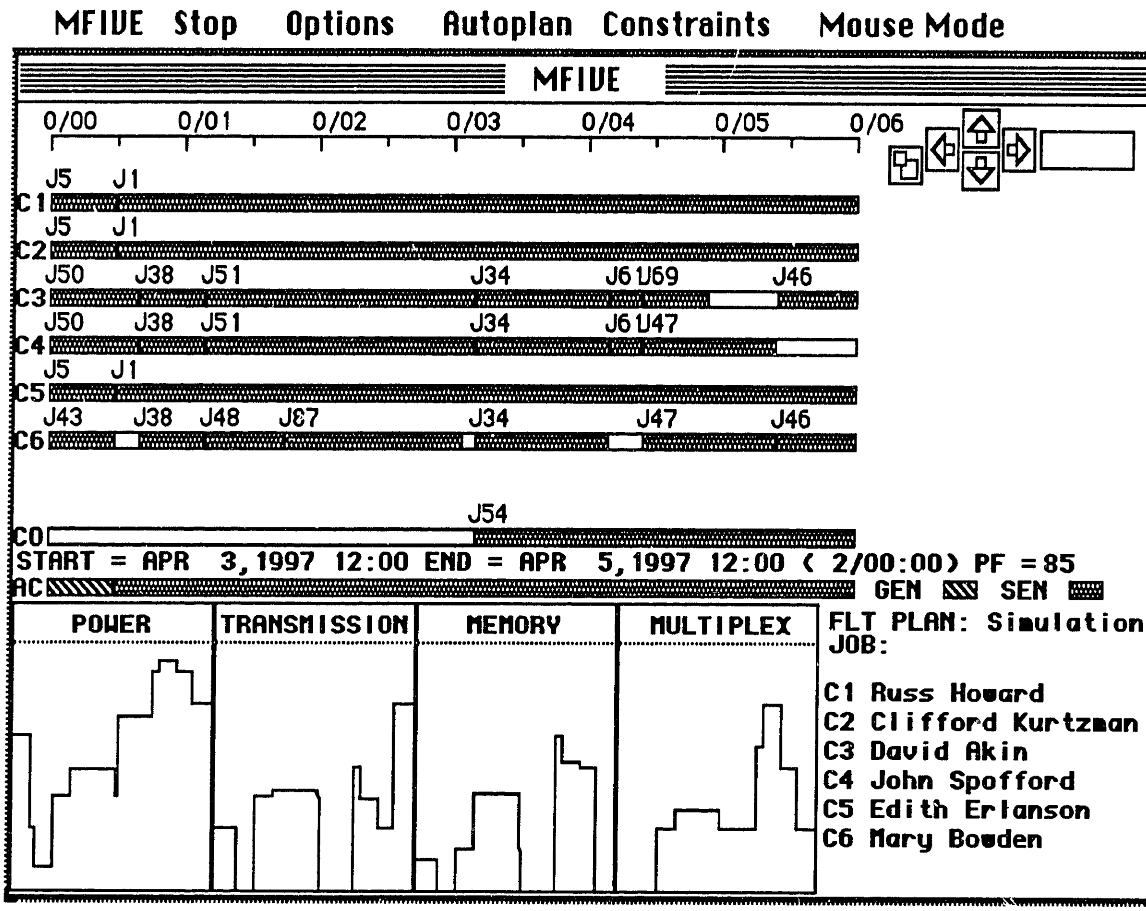


Figure 5.28A: Hours 0 - 6

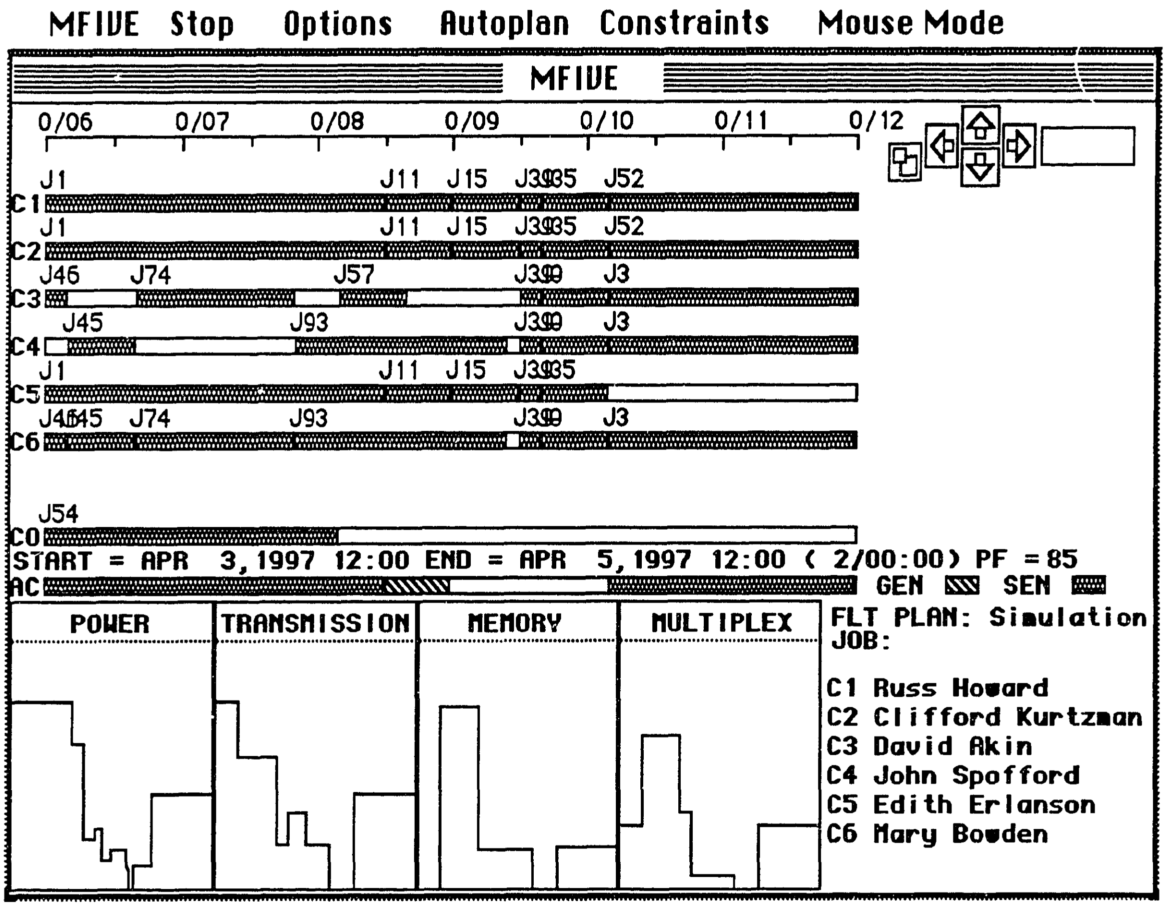


Figure 5.28B: Hours 6 - 12

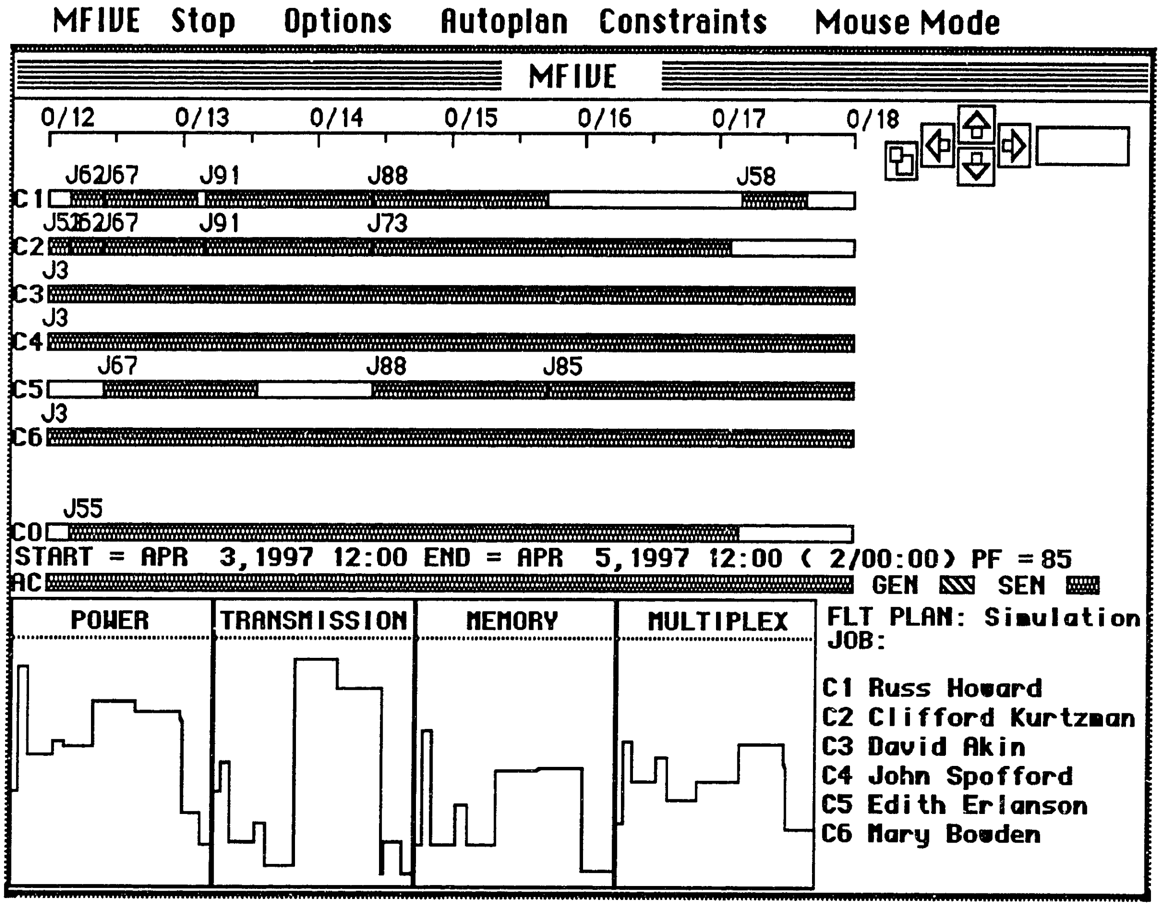


Figure 5.28C: Hours 12 - 18

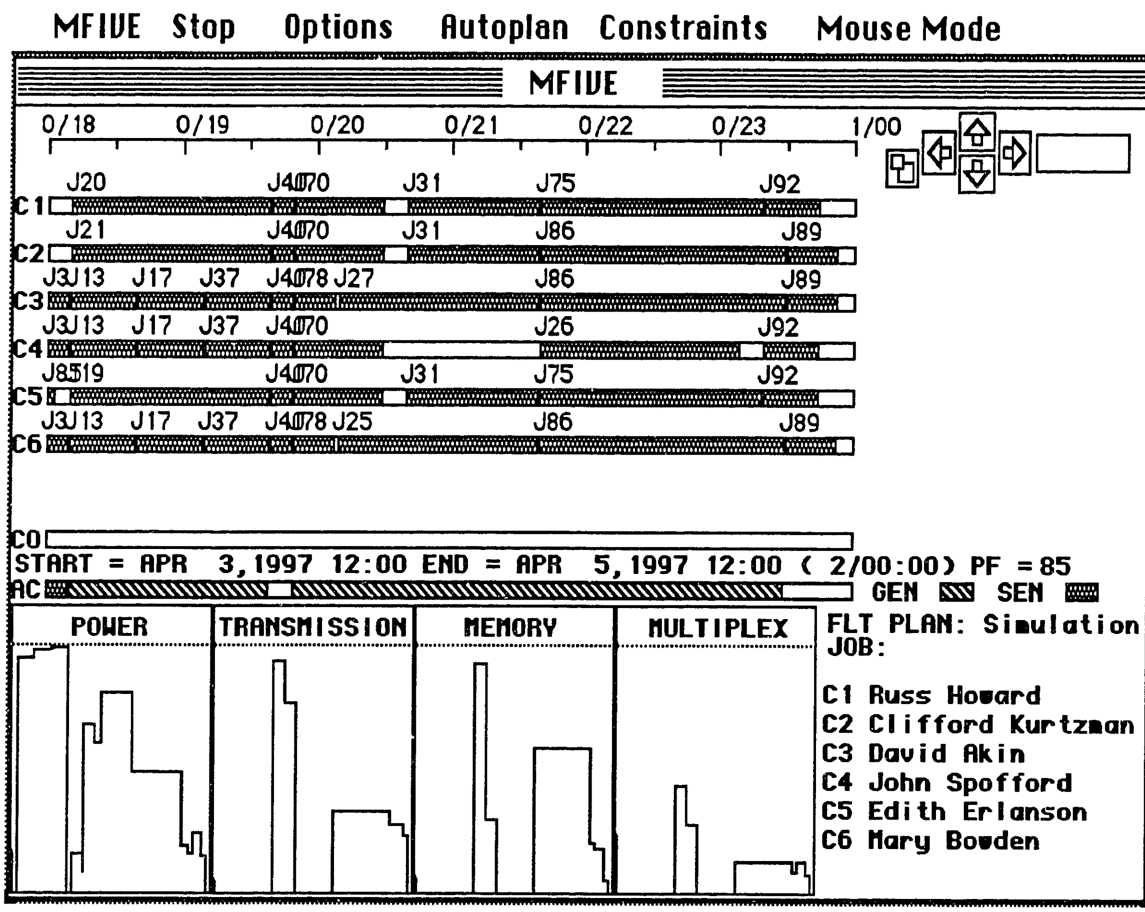


Figure 5.28D: Hours 18 - 24

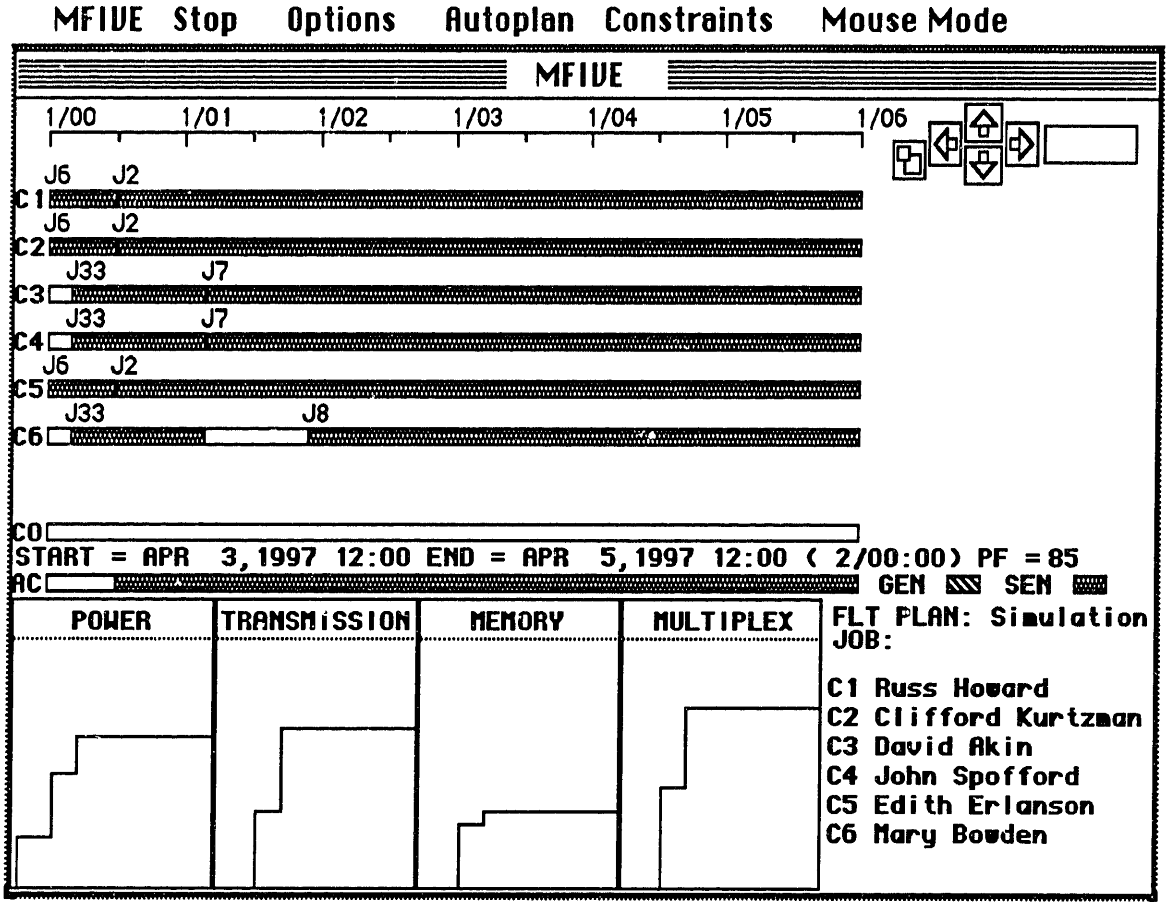


Figure 5.28E: Hours 24 - 30

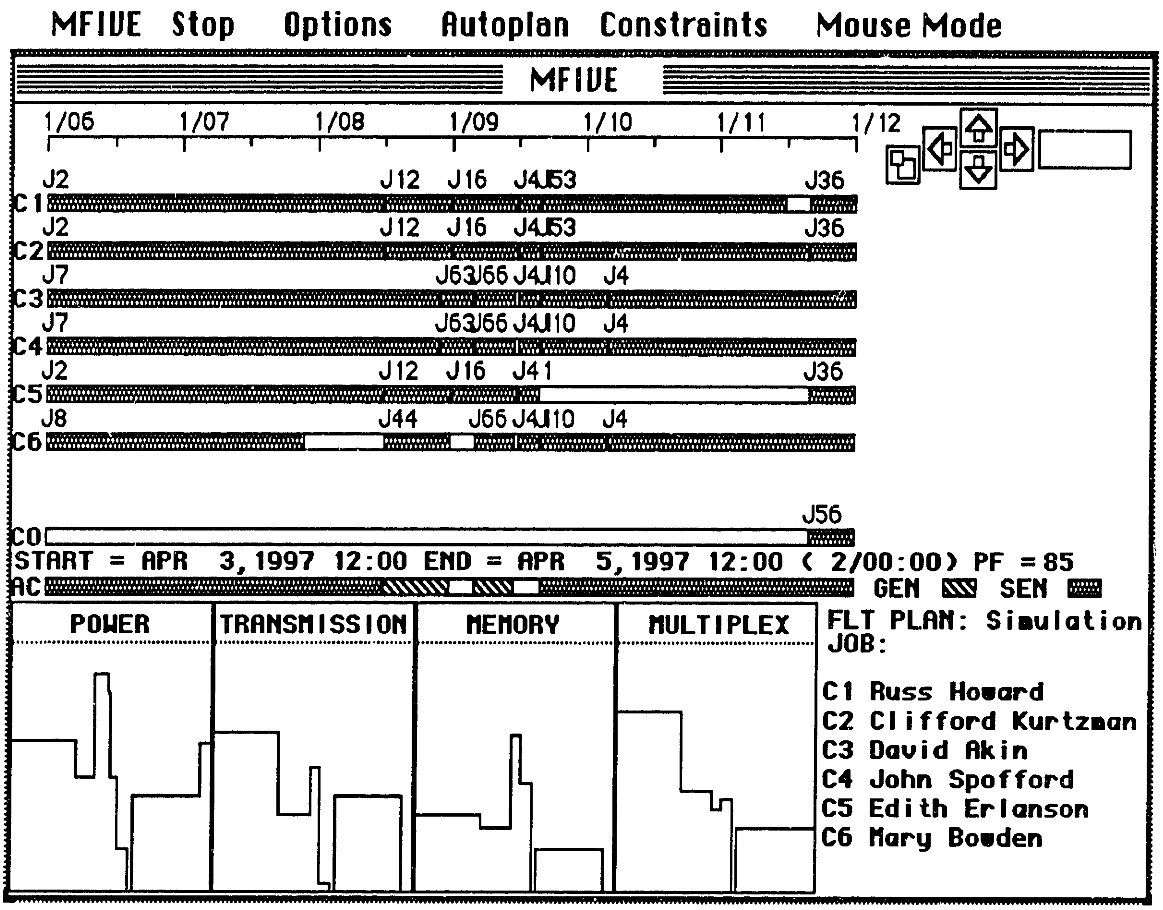


Figure 5.28F: Hours 30 - 36

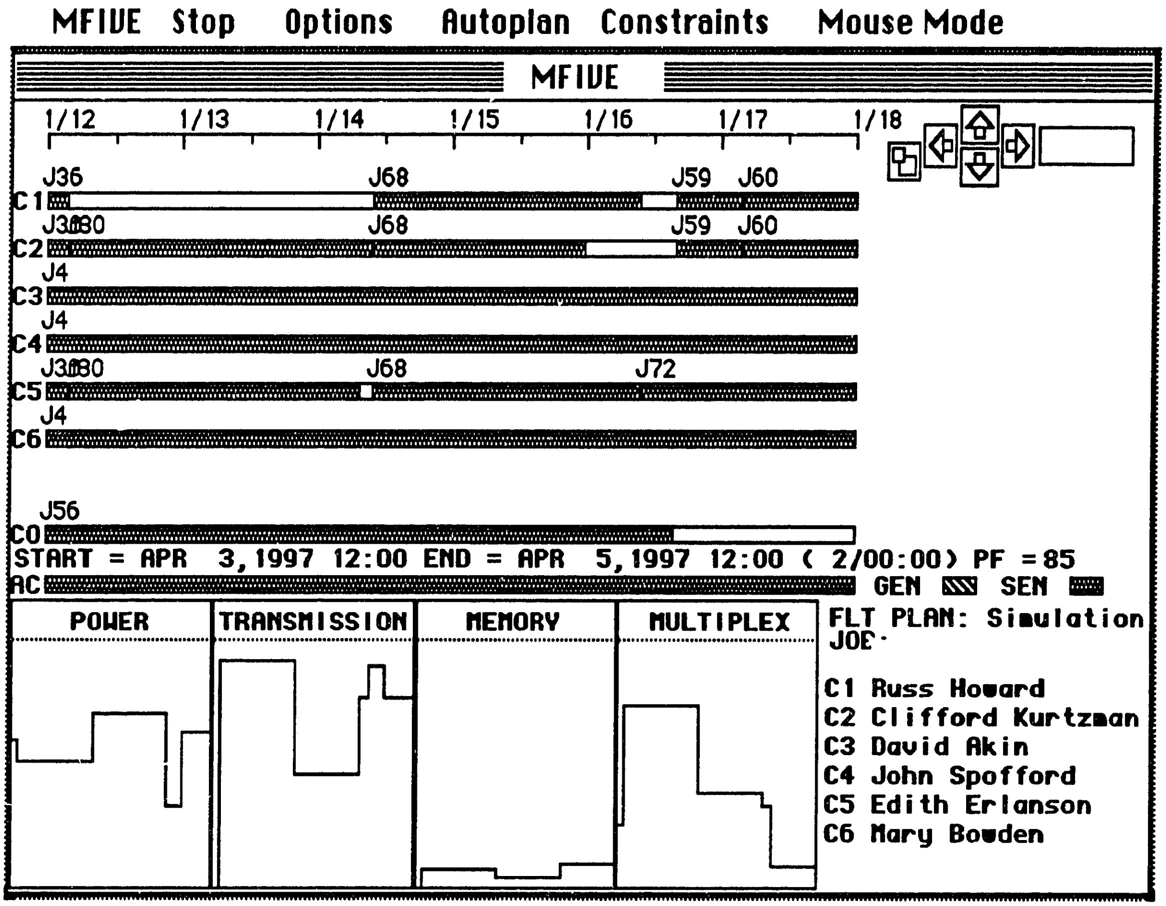


Figure 5.28G: Hours 36 - 42

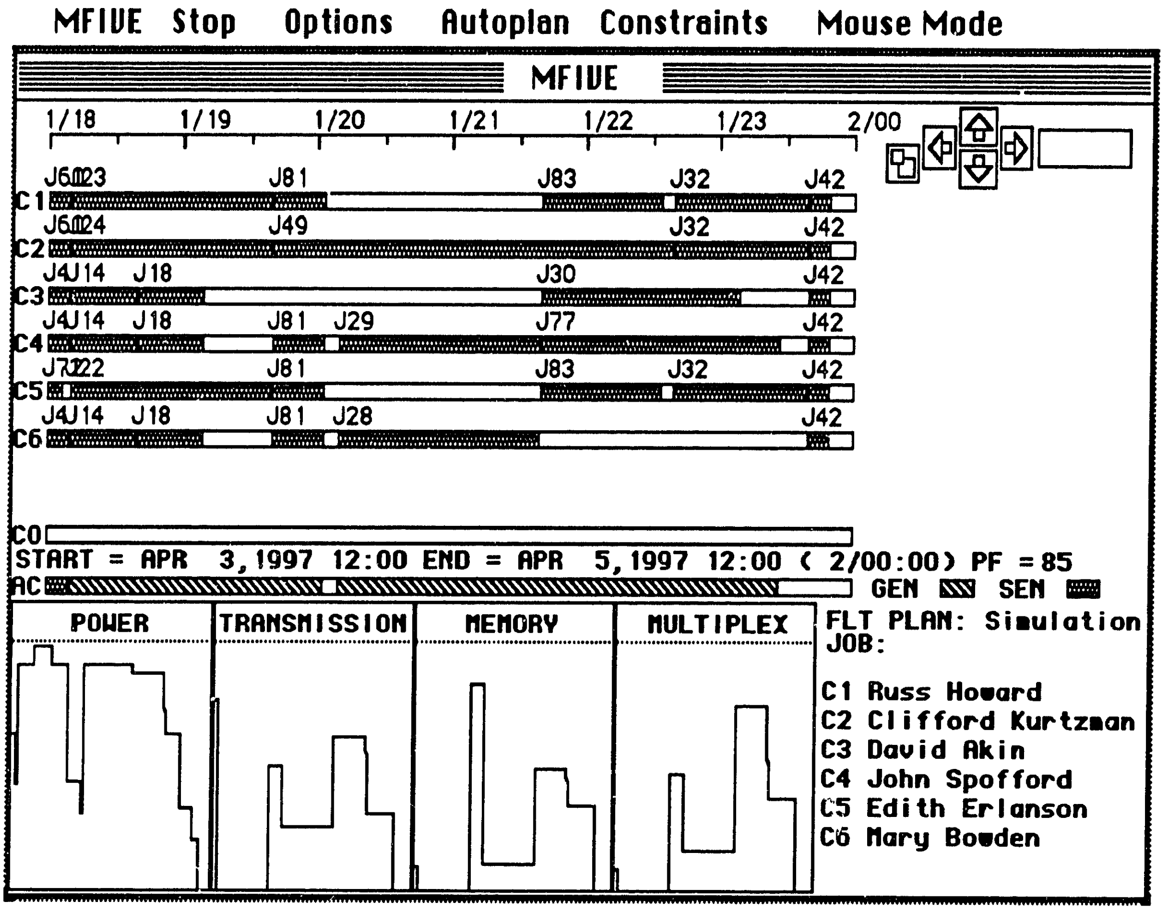


Figure 5.28H: Hours 42 - 48

Section 6: The MFIVE Space Station Crew Activity Planner

In order to devise, test, and implement the algorithms developed in this thesis, a software tool, named the MFIVE Crew Activity Planner, was developed. The MFIVE software is implemented on a Apple Macintosh computer using the APL computer language. MFIVE provides a convenient and user friendly interface for building, solving, and displaying scheduling problems, as well as for investigating the features which will be necessary to eventually provide a real-time scheduler for use on a space station. Figure 6.1 presents a summary of computer based systems (discussed in Section 3 and in this section) for performing planning and scheduling functions similar to crew activity planning. From the start, MFIVE was intended to complement, rather than compete with, other projects, such as KNEECAP [Mogilensky, et al., 1983] and AMPASES [Jakubowicz, 1985]. These programs are implemented on much more sophisticated (and costly) LISP machines.

There are several other software efforts which have designed scheduling software for personal computers [Freedman, 1985; Stevens, 1987], but these projects usually are limited to standard PERT/CPM techniques which cannot accomodate the complexities present in space station scheduling, such as resource constraints. Available scheduling systems include, among others, the *Total Project Manager* from Harvard Software; *Pac III* from AGS Management Systems; *PMS-II* from North America Mica; *MicroGANTT* from Earth Data; *MacProject* from Apple Computer; *Micro Planner Plus* from Micro Planning International; and *AEC Information Manager* from AEC Management Systems Inc.

MFIVE models the core features of the space station scheduling environment. While MFIVE makes many simplifications of space station operations, these instances do not fundamentally alter the nature of the scheduling problem. For example, in the description of a space station experiment, MFIVE has a parameter describing the amount of power (in Watts) used by the experiment. In a more realistic scenario, one would have to specify many parameters, such as nominal power, peak power, periods of operation (e.g., day-night cycles and duty cycles), power type (AC/DC), and voltage, etc. The inclusion of this information would not fundamentally change the nature of the scheduling problem: one would just have to perform more complicated checks to determine when it would be feasible to schedule the experiment. There is no reason why this type of information could not be added to the MFIVE system. However, including this type of data now would only slow program operation and consume substantial programming time, while not really adding anything "new" to the system; this is not in keeping with the goals of this study.

Figure 6.1: Computer Based Systems for Planning and Scheduling

(not an exhaustive listing)

Advanced Expert/Planning Systems For
Space Shuttle or Space Station Scheduling

MFIVE
PLAN-IT
AMPASES
KNEECAP
SSES

Expert Systems for Space Related Functions

EMPRESS
Deviser
ESSOC
NAVEX
OpSim

AI Planning Systems

NOAH
STRIPS
MOLGEN
NUDGE

Expert Systems for Job Shop Scheduling

ISIS
MASCOT
OPAL

NASA Software for Space Scheduling

Crew Activity Scheduling Program (CASP)
Fast Automated Scheduling Technique (FAST)
Manned Activity Scheduling System (MASS)
Viking Lander Sequence of Events Scheduler (LSEQ)
Crew Activity Planner (CAP)
Timeline Program (TLP)

Scheduling Software for Personal Computers

The Total Project Manager
PAC III
PMS-II
MicroGANTT
MacProject
Micro Planner Plus
AEC Information Manager

It was originally anticipated that a LISP/PROLOG-like approach would be desirable for generating English language rules on which to base the scheduler. Further investigation, however, proved otherwise. A LISP-like approach is very well suited to providing some user interfaces (such as explanations of reasoning chains), and for doing things like constraint satisfaction via rule checking and inference. The process of finding an optimal schedule, however, involves highly mathematical algorithms, which are usually not well suited to English-based rules. There is, of course, no reason that these algorithms cannot be implemented in LISP; it is just that it is unnecessary (and often more complicated) to do so.

APL was therefore chosen as the implementation language for MFIVE. The high level features of APL allow very quick program testing and development. APL is a highly mathematical computer language which allows easy vector/matrix manipulation, and is well suited to implementing operations research algorithms. Being interpreted, however, APL is rather slow compared to some other compiled computer languages. It is not anticipated that APL would be the language of choice for an operational system; it is, however, well suited to the goal of algorithm development and testing.

MFIVE is, at the highest level, organized into three modules (Figure 6.2): the PRIME module serves as a data base management system, allowing the user to enter and modify all data pertinent to the crewmembers, the jobs they perform, the tools used, the stowage layout of the space station, and the assignment of crew and jobs into organized flight plans; the SCHED module allows the interactive and automatic scheduling of a flight plan; and the SEARCH module performs searches for tools which are difficult to locate. The following sections provide an overview of the current operation of the MFIVE program. The actual details of MFIVE operation are presented in Appendix B, the MFIVE Users Guide. At the end of each section, suggestions are made indicating ways in which future developments might enhance the operation of the system.

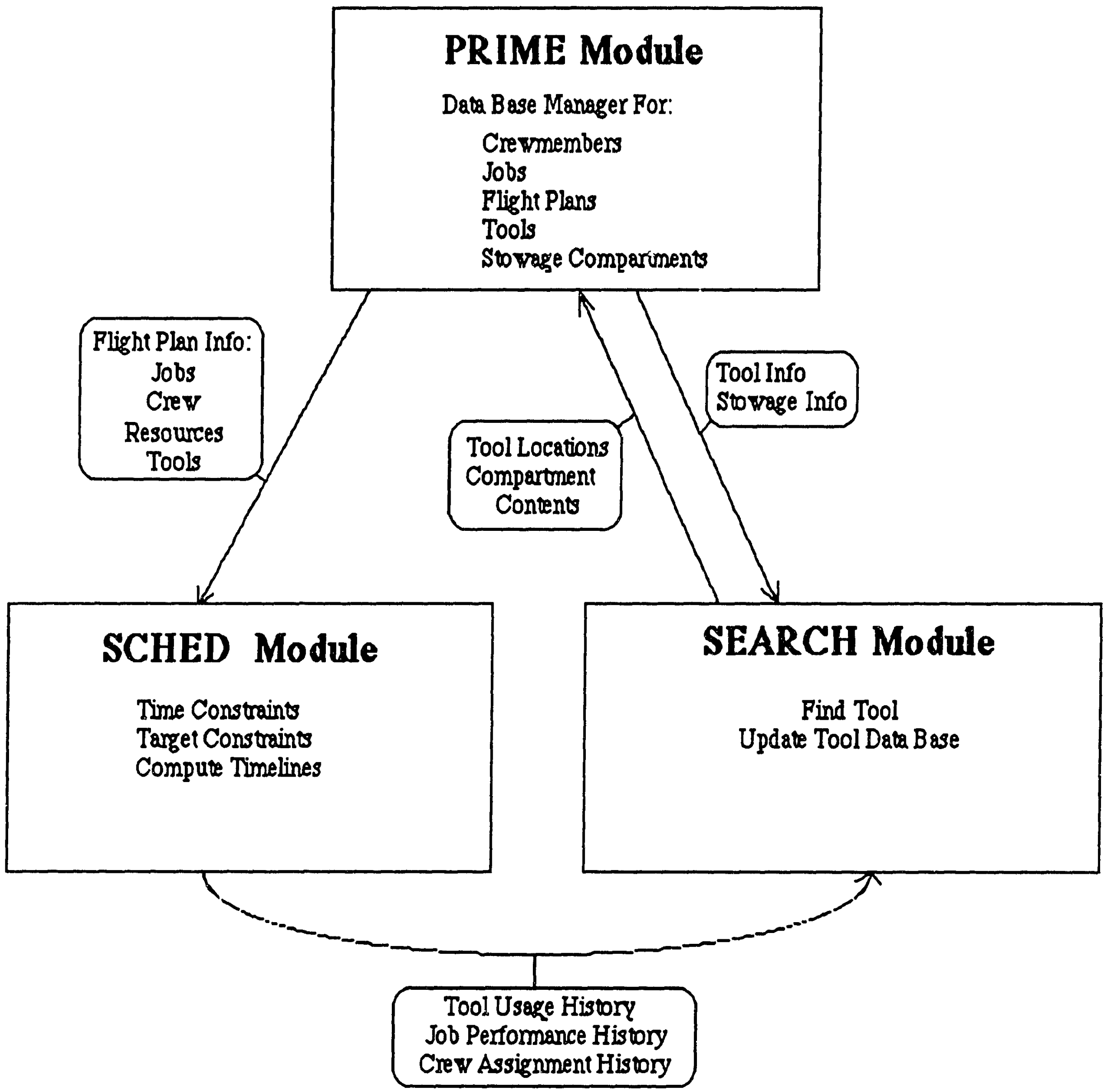


Figure 6.2: MFIVE Organizational Structure

6.1 The Data Base Management System

The PRIME module of the MFIVE system presents the user with five icons (Figure 6.3) which allow access to the various objects (crewmembers, jobs, flight plans, tools, and stowage bins) in the data base management system. The sixth EXIT icon allows the user to terminate MFIVE operation and return to the APL environment. For example, selecting the icon labelled CREW will allow the user to call up a generic Crew Information Form (Figure 6.4). With this form, the user can add a new crewmember to the database, or inspect, add, modify, or delete information pertaining to some crewmember already known to the system.

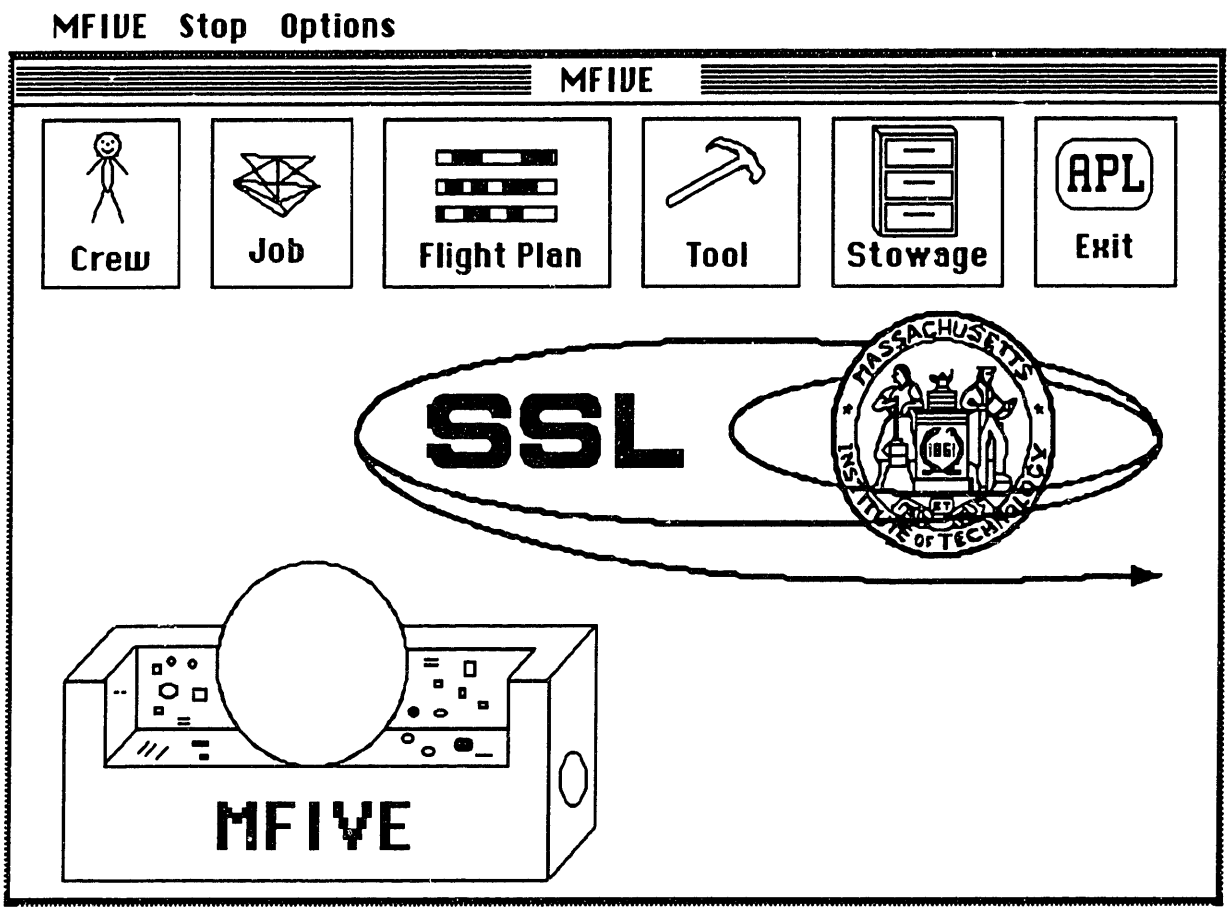


Figure 6.3: MFIVE Data Base Management System

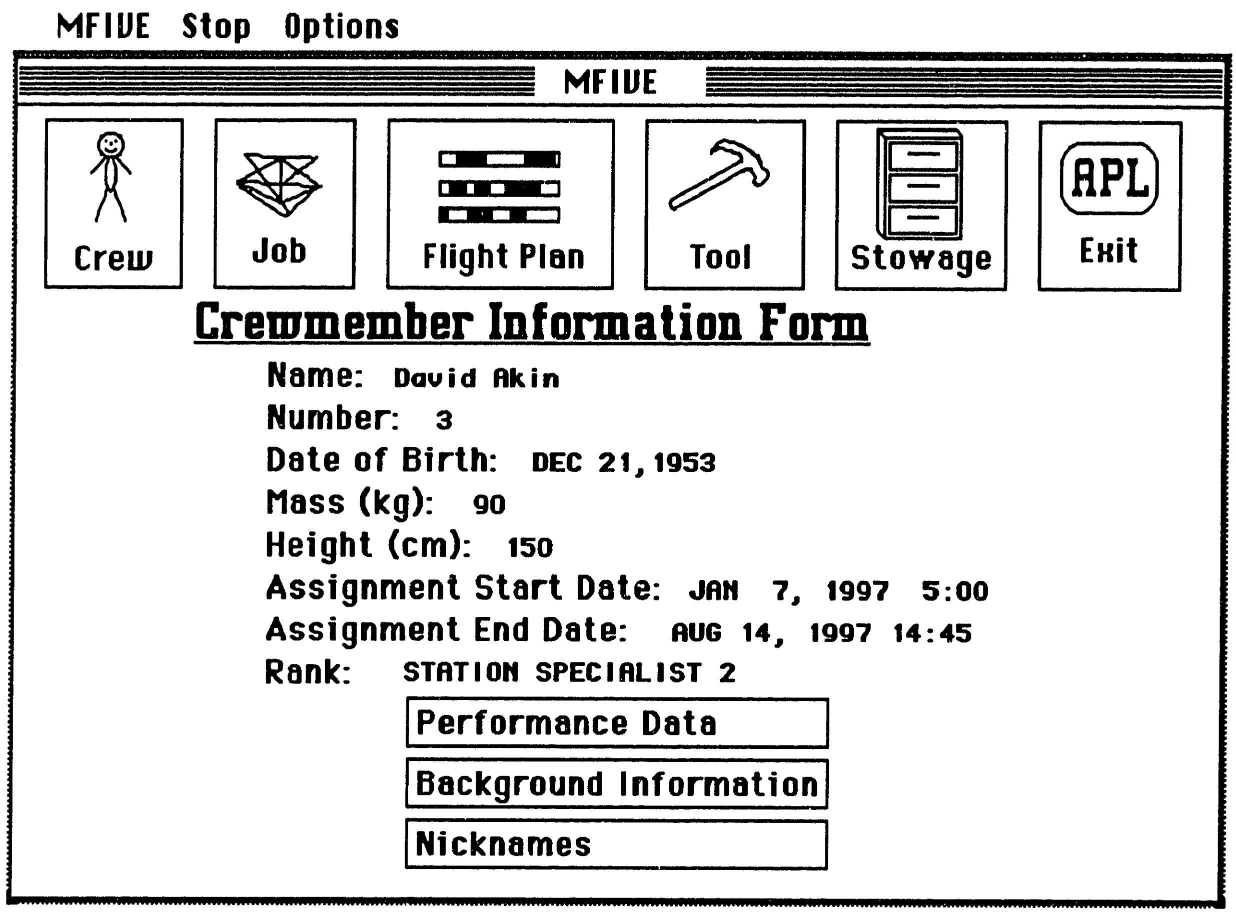


Figure 6.4: The Crewmember Information Form

As Figure 6.4 shows, some of the data applicable to a crewmember is displayed directly on the screen (e.g. date of birth), in which case the information can be modified by clicking on the current entry, and then entering a new value in response to a prompt by MFIVE. Other information (e.g. crewmember background information) is initially hidden, and is brought to the screen for revision by clicking on the applicable box (Figure 6.5). The next sections outline the details of the various information forms.

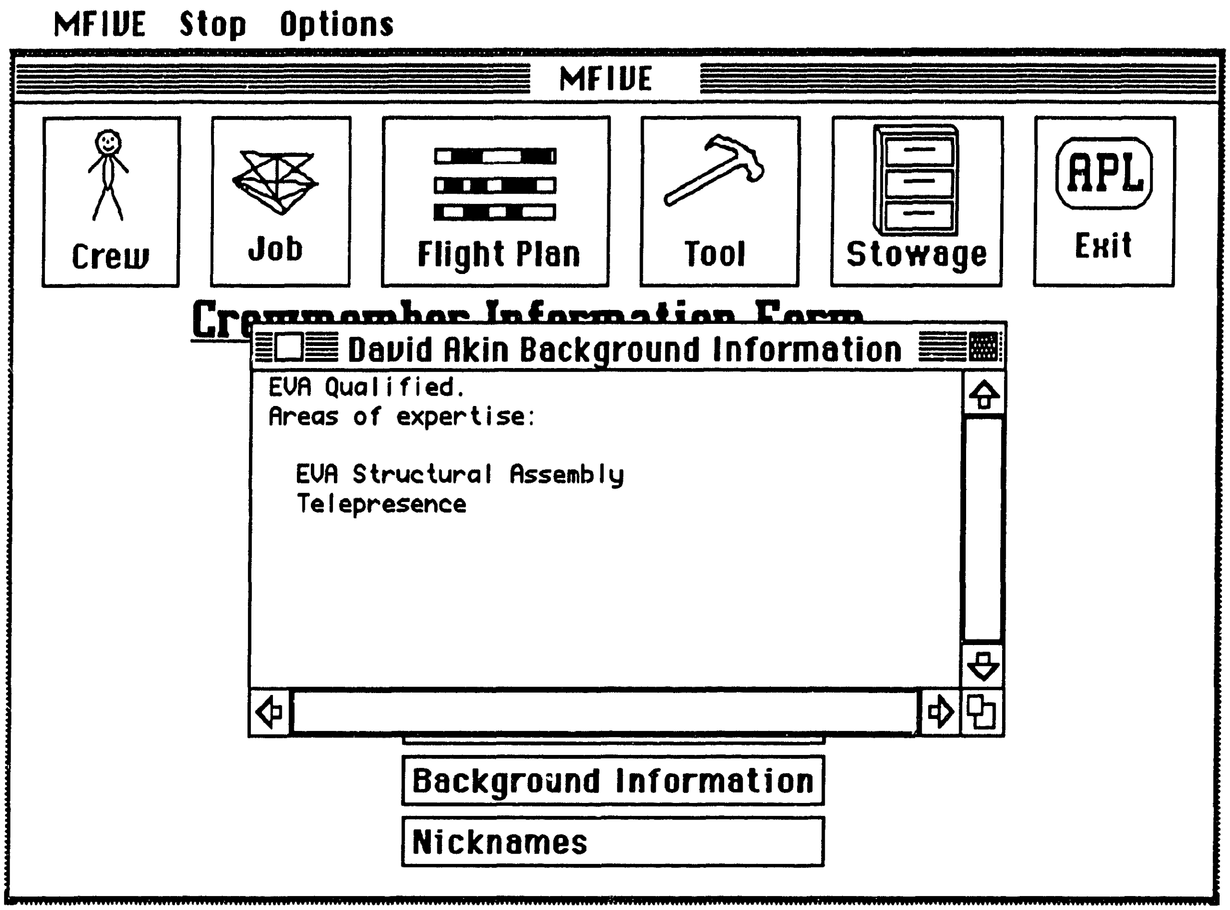


Figure 6.5: Background Information for Crewmember Akin

6.1.1 The Crewmember Information Form

The Crewmember Information Form maintains information pertinent to each crewmember known to the MFIVE system. For each crewmember, data is maintained on the crewmember's height, mass, date of birth, space station assignment dates, rank (e.g. Space Station Specialist) and alternate "nicknames" by which MFIVE will recognize the crewmember. Also accessible through the information form is background information on the crewmember (Figure 6.5), and the performance time (Figure 6.6) of the crewmember on each job cataloged by the system (see Section 6.1.3).

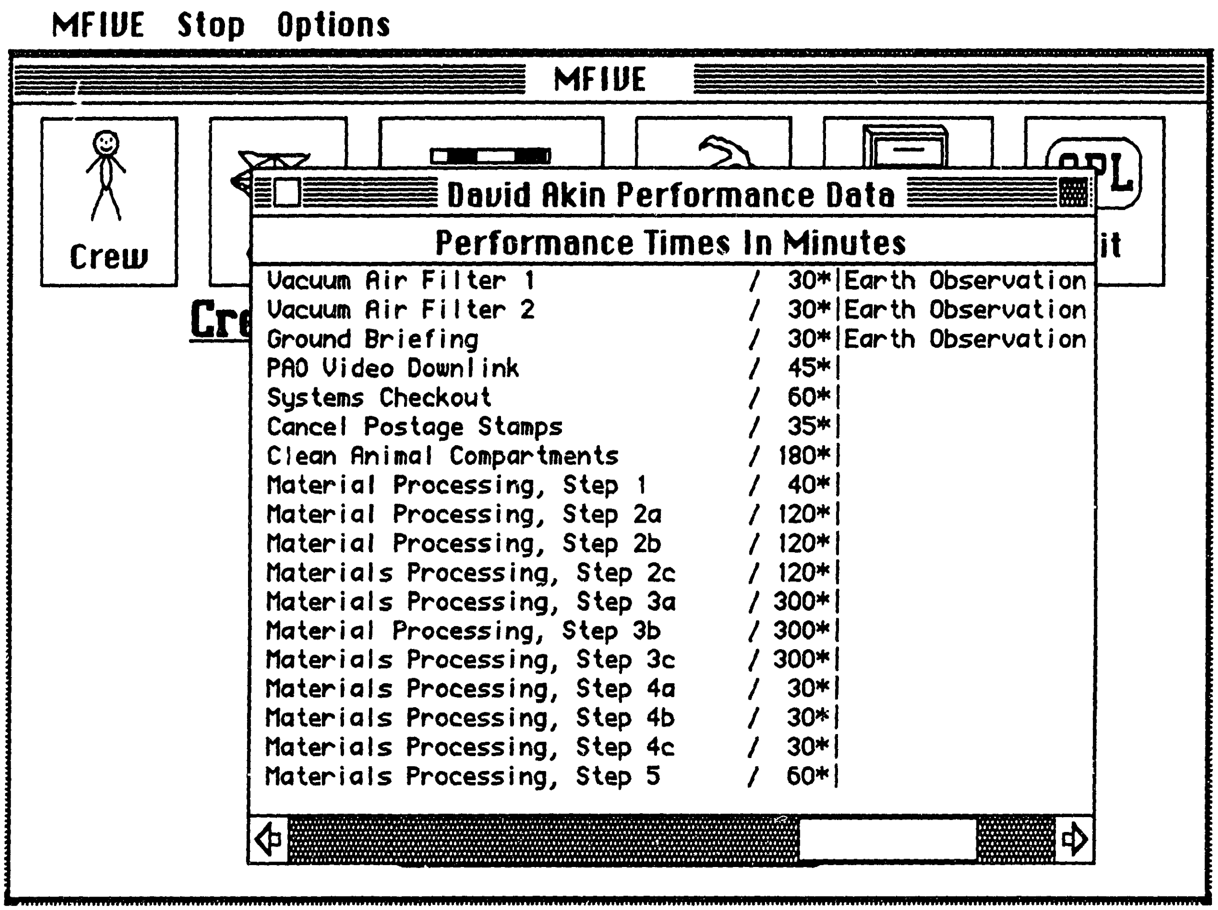


Figure 6.6: Job Performance Times for Crewmember Akin

6.1.2 The Job Information Form

The Job Information Form (Figure 6.7) maintains the characteristics of each of the jobs known to MFIVE. The number of crewmembers needed to perform the job is listed, as well as the "default" performance time of the job, which is the time taken by each crewmember to perform the job unless specific overriding information is provided (via the performance time windows on the Crewmember or Job Information Forms, Figures 6.5 and 6.8). This might occur, for example, if a particular crewmember is more highly trained on a particular job and can hence perform it faster.

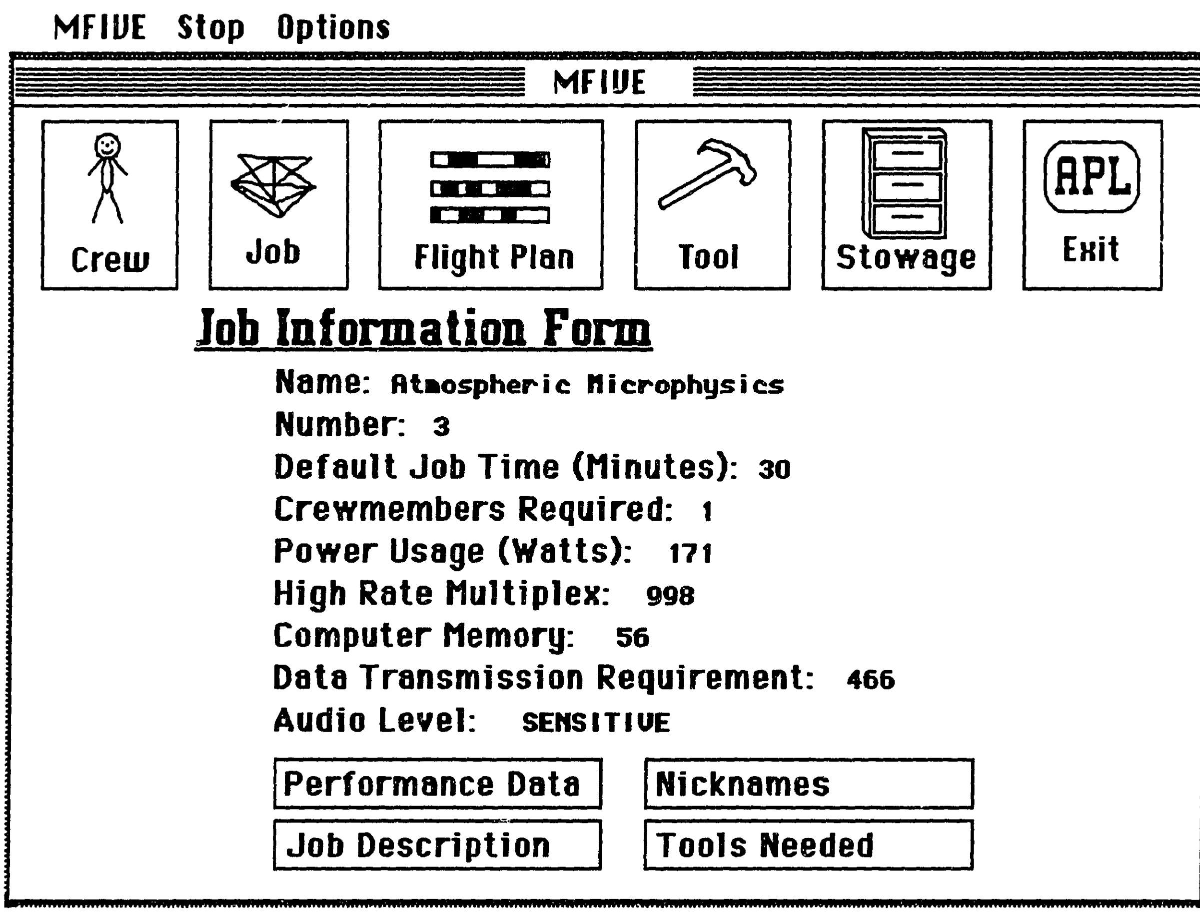


Figure 6.7: The Job Information Form

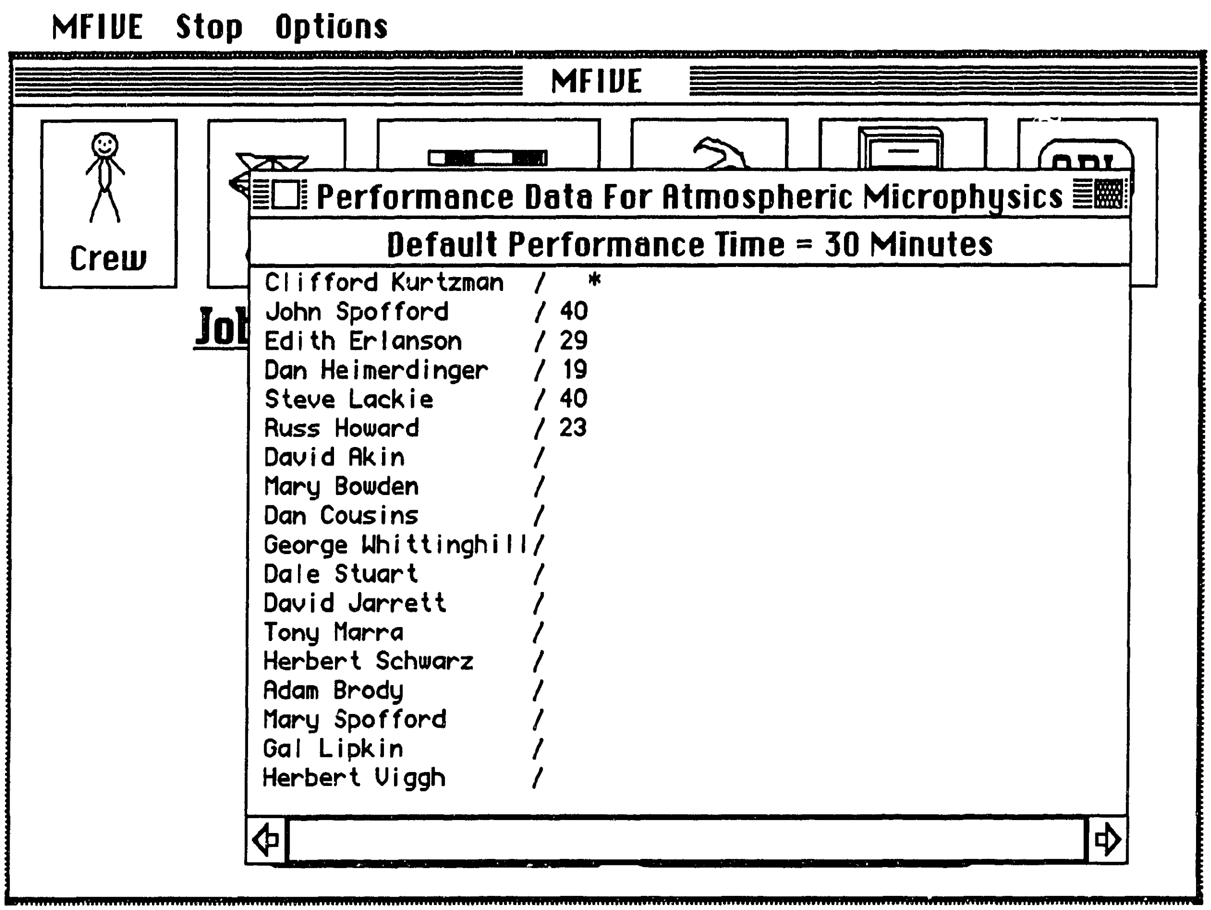


Figure 6.8: Crewmember Performance Times for the Job Atmospheric Microphysics

Also on the Job Information Form are the amounts of each of four resources used by the job: power, data transmission, high rate multiplex, and computer memory. These resources were chosen to be representative of the types of resources which will be used aboard the space station. The current implementation of MFIVE assumes that the rate of resource usage is constant during the duration of the job. An audio level resource is also included on the form. Each job is classified as being either noise and vibration sensitive, neutral, or generating. No job which is noise sensitive can be performed while another job which is noise generating is being performed.

By selecting the appropriate rectangles on the screen, information can be called up to show a description of the job (or instructions for performing it) (Figure 6.9), alternate "nicknames" for the job (Figure 6.10), and a listing of the names and quantities of the tools which are needed to perform the job.

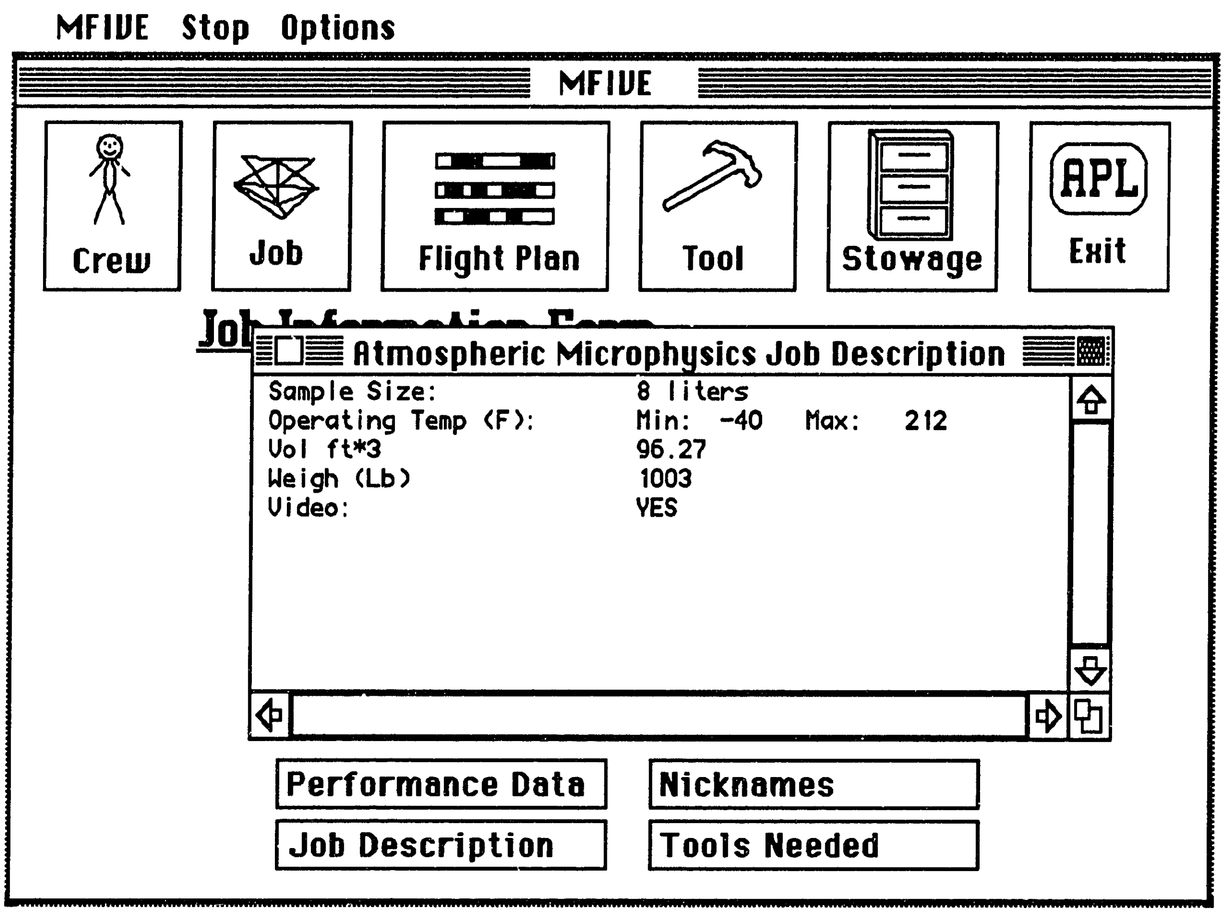


Figure 6.9: Job Description for Atmospheric Microphysics

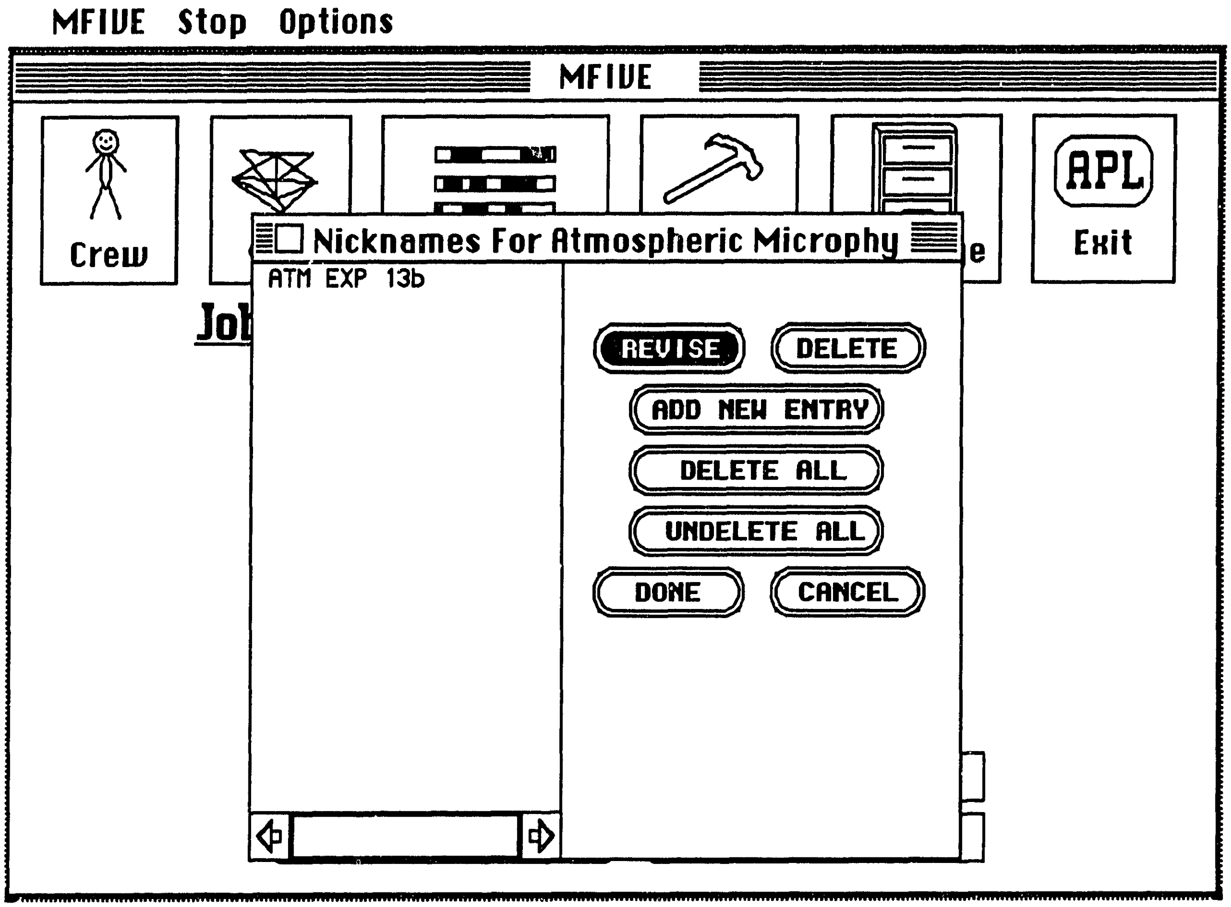


Figure 6.10: Alternate Names for Atmospheric Microphysics

6.1.3 The Flight Plan Information Form

A Flight Plan (Figure 6.11) allows groups of jobs to be assembled into units for scheduling by selected crewmembers. A user can enter the start and end dates of a flight plan, which specify the interval during which all the jobs must be scheduled. Limits can also be set on each of the four resources tracked by the scheduler. Selecting the appropriate rectangle on the information form, crewmembers (Figure 6.12) and jobs (Figure 6.13) can be assigned to the flight plan. Once all the relevant parameters of a flight plan and it jobs and crewmembers have been specified, selecting the "Schedule" option will route the user to the scheduler so that a timeline can be planned out (see Section 6.2).

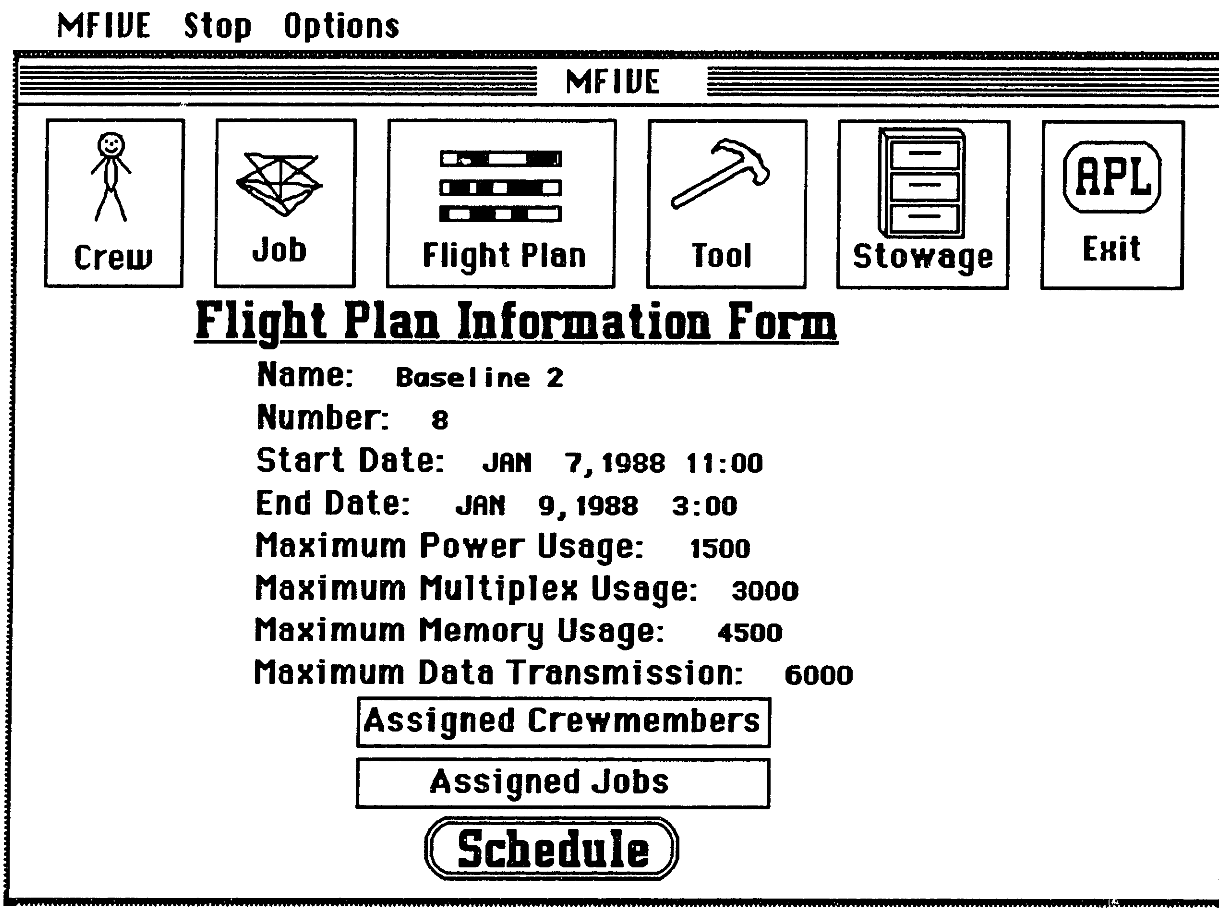


Figure 6.11: The Flight Plan Information Form

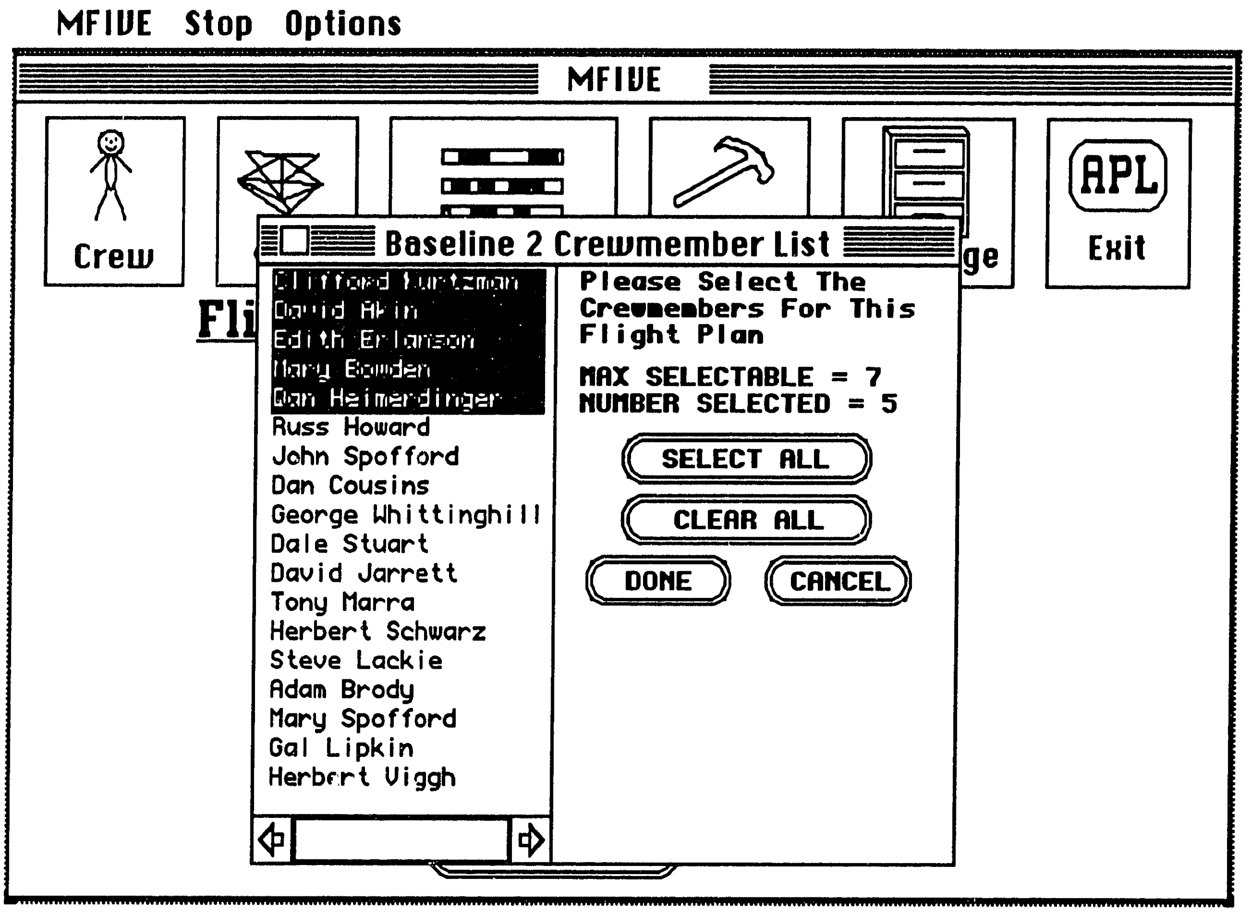


Figure 6.12: Selecting Crewmembers for a Flight Plan

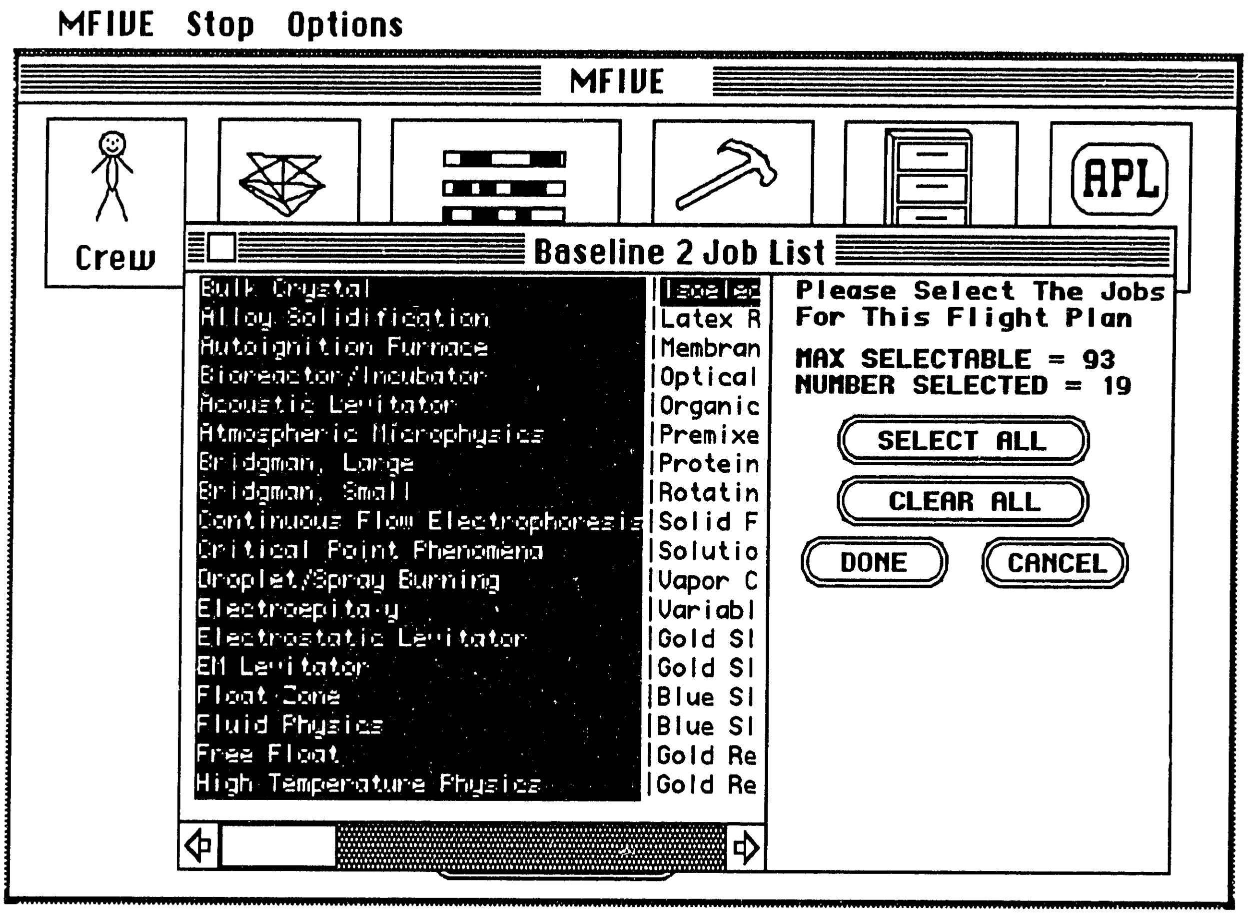


Figure 6.13: Selecting Jobs for a Flight Plan

6.1.4 The Tool Information Form

The Tool Information Form (Figure 6.14) maintains data on equipment stored aboard the space station. The form indicates the number of copies of each tool in stock, as well as the effective volume of each tool. (For example, a tool with an effective volume of 10 liters cannot fit in a compartment with a smaller effective volume.) The Default Locations and Status area on the form can be selected to show where each copy the tool is supposed to be located, as well as the status of that copy (Figure 6.15). The form can access usage instructions for each tool (Figure 6.16) and the jobs for which the tool us used (Figure 6.17). Selecting the "Search" option will engage the searcher to aid in finding missing tools (see Section 6.3).

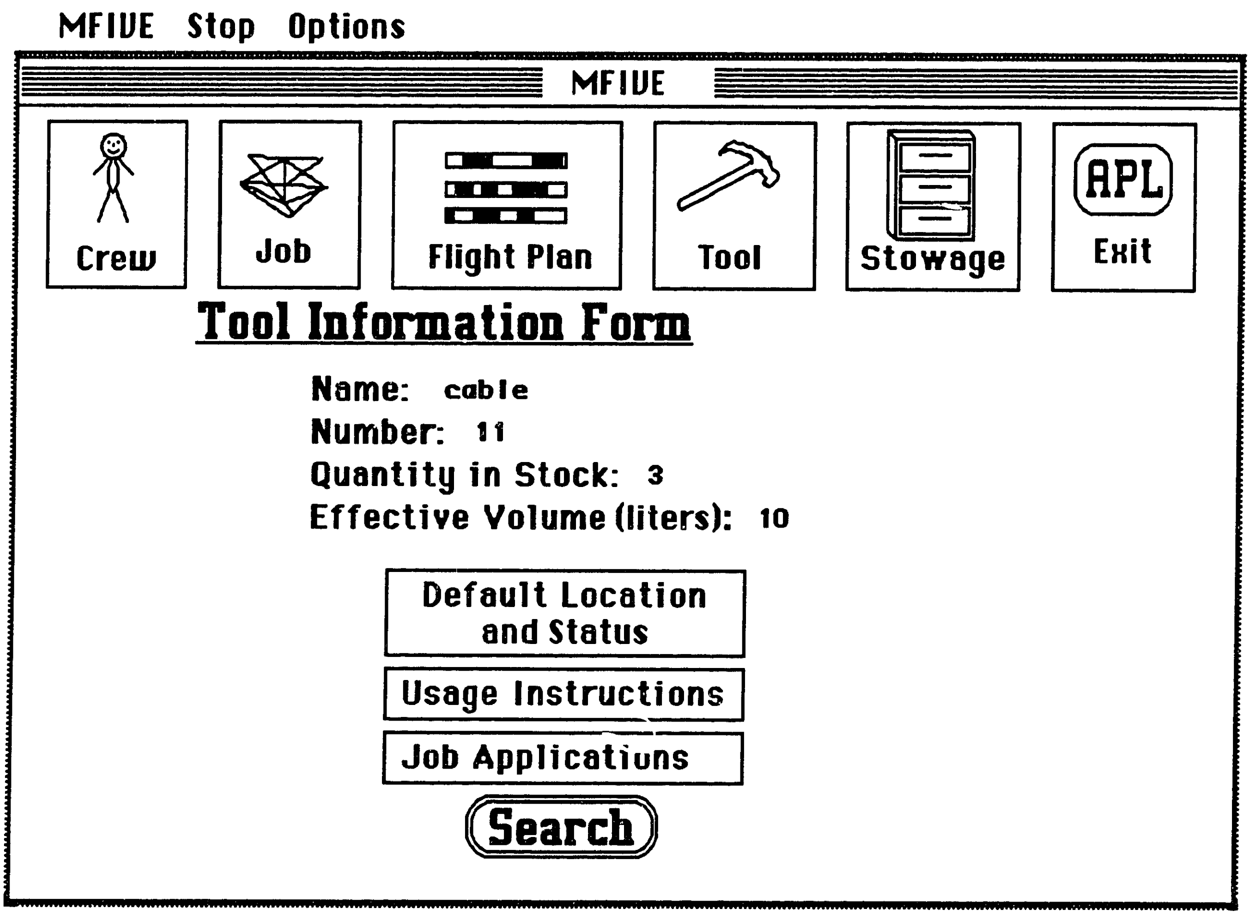


Figure 6.14: The Tool Information Form

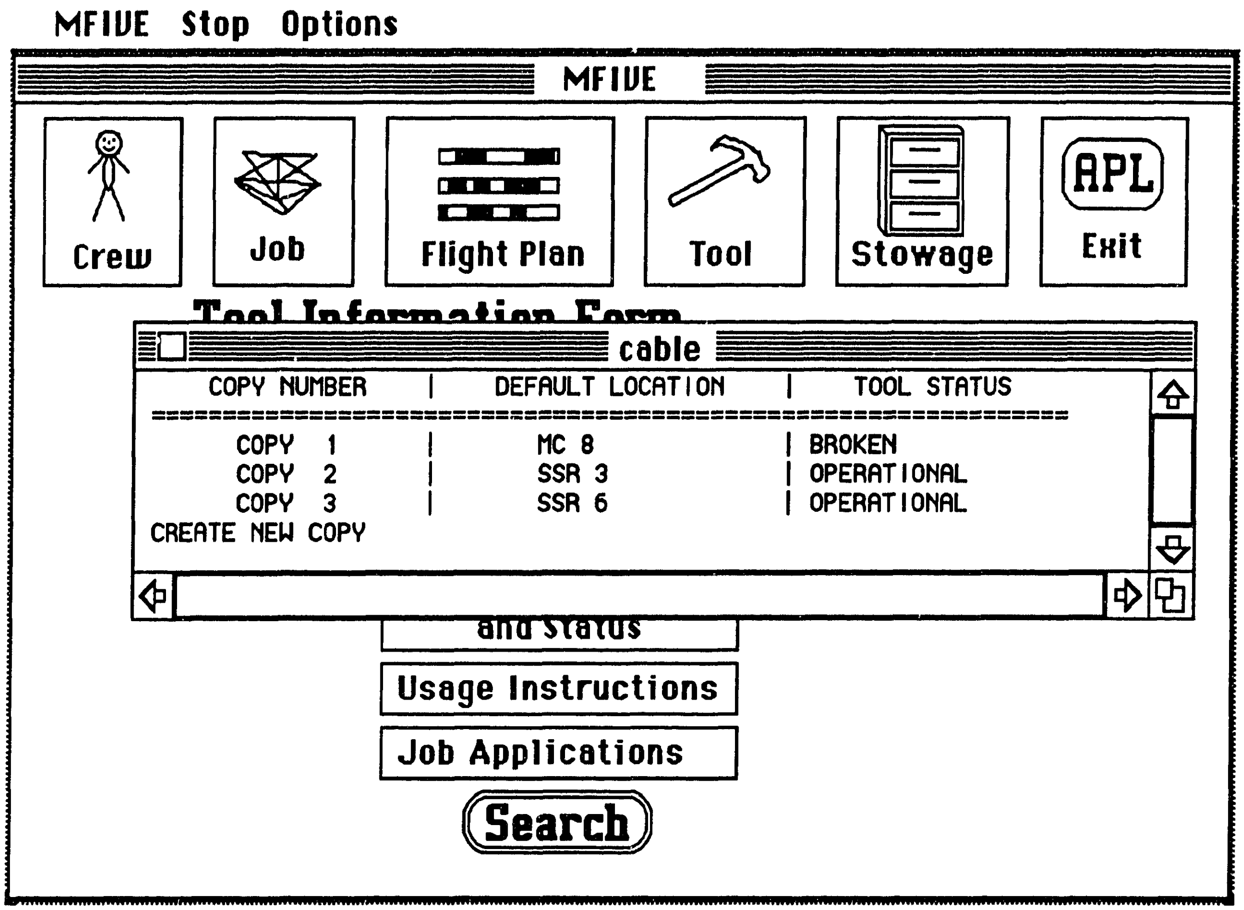


Figure 6.15: Default Locations and Status for the Tool "Cable"

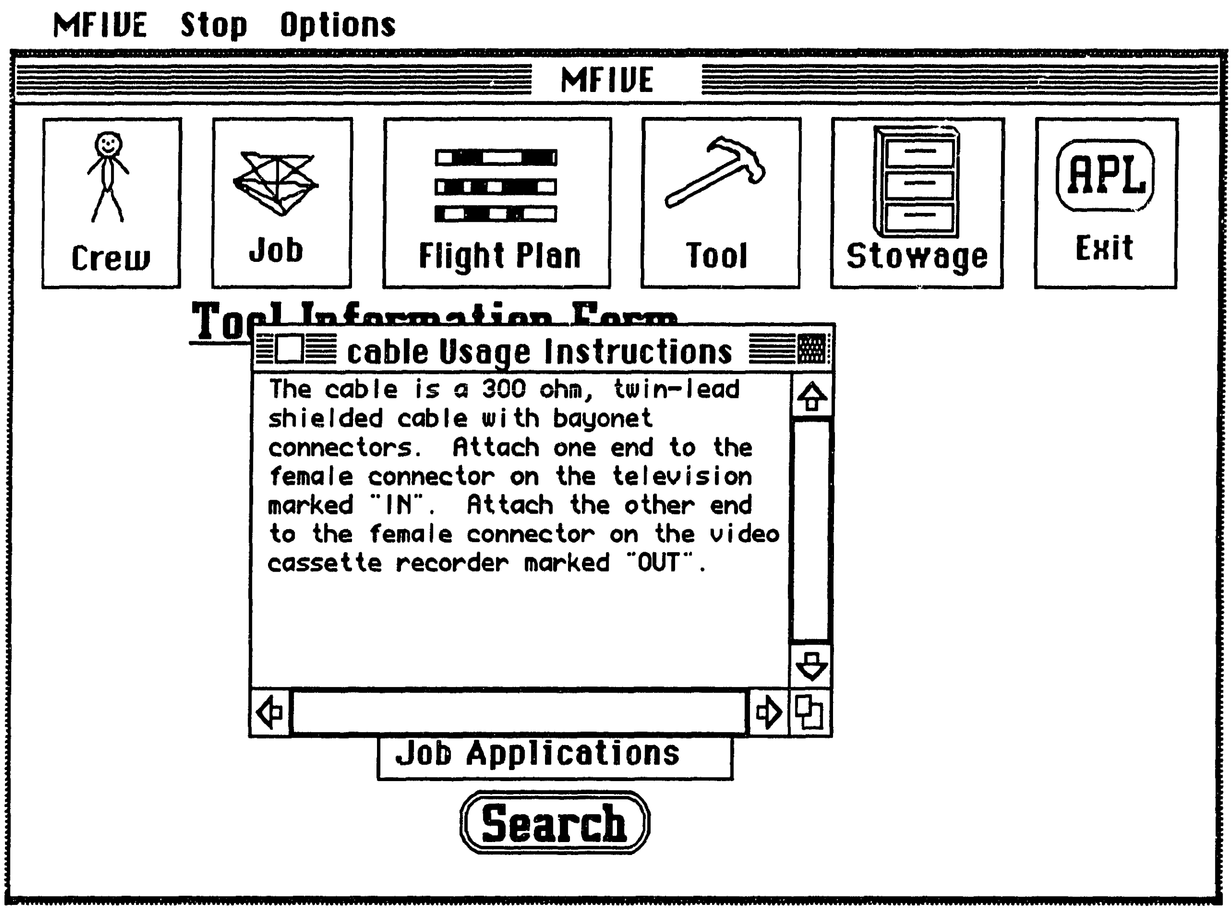


Figure 6.16: Usage Instructions for the Cable

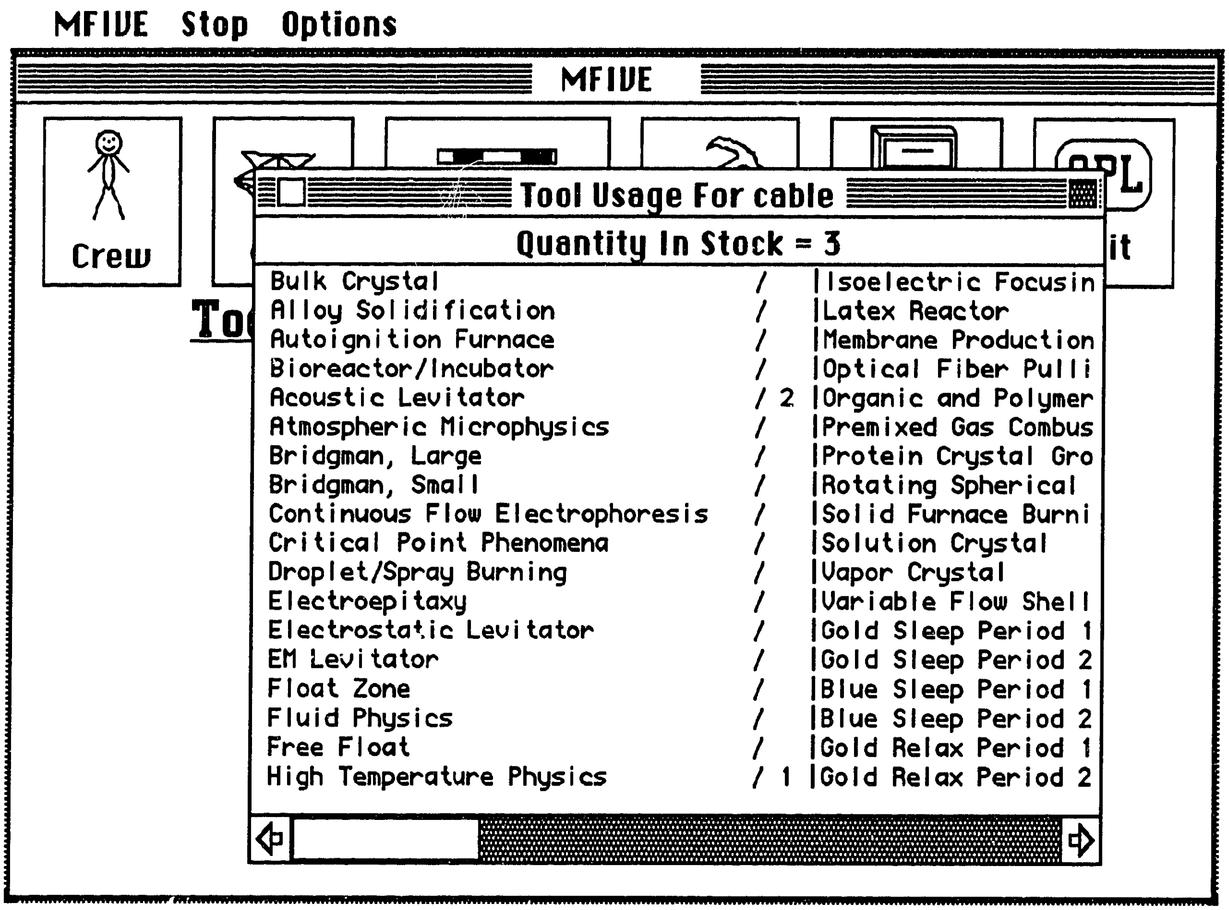


Figure 6.17: Jobs Using the Cable

6.1.5 The Stowage Information Form

The Stowage Information Form (Figure 6.18) shows the effective volume of each compartment, the space station module in which that compartment is located, and the compartment's location within that module (in cylindrical coordinates). The contents of each compartment can also be viewed by selecting the Default Contents rectangle (Figure 6.19).

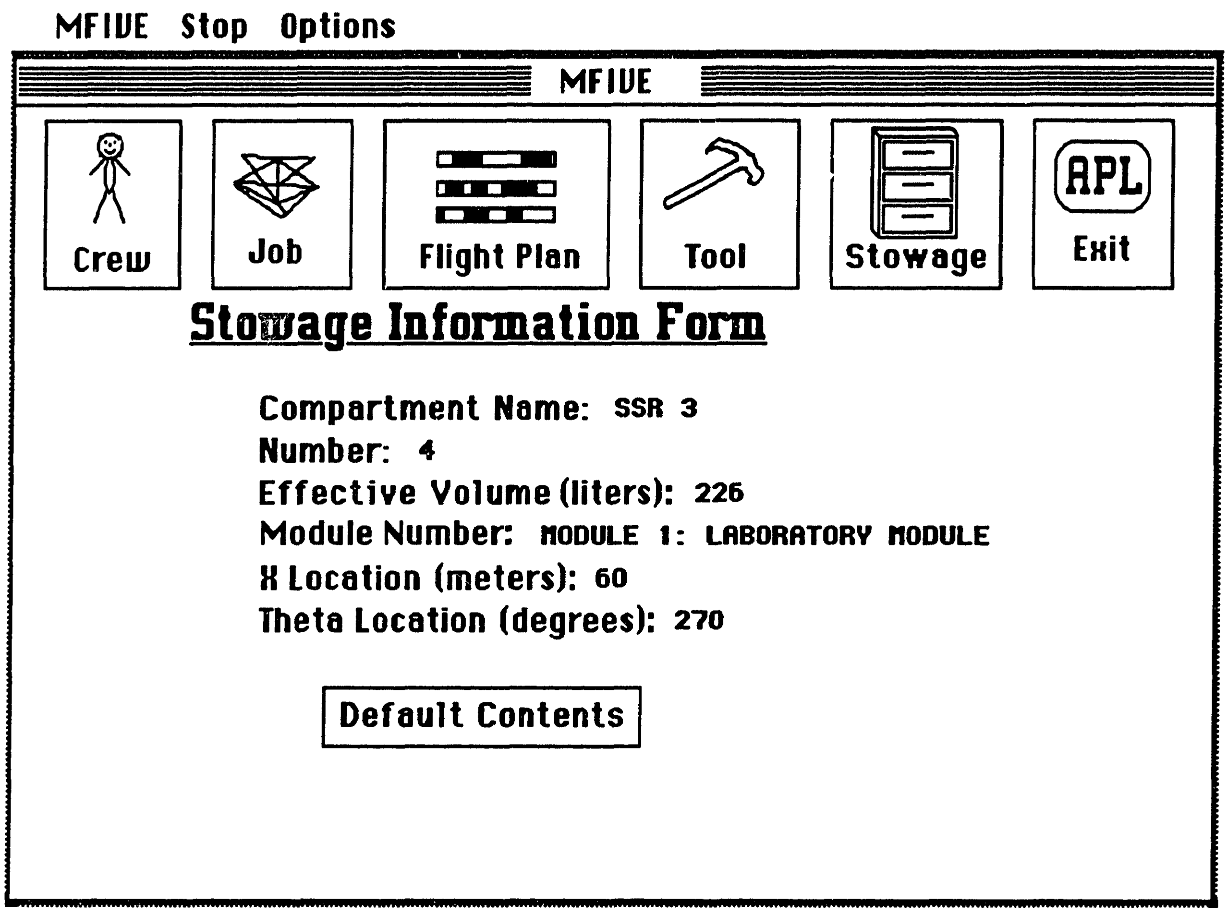


Figure 6.18: The Stowage Information Form

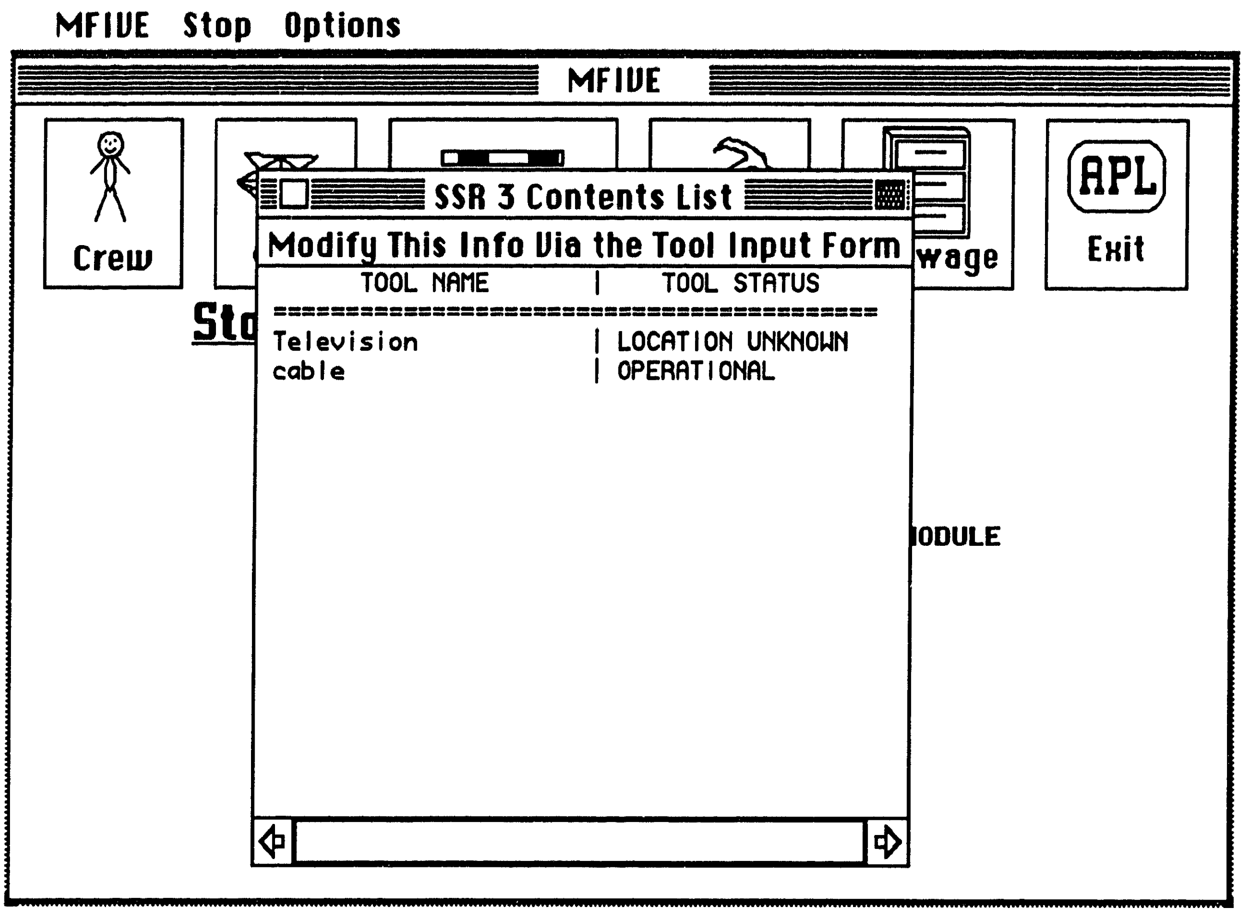


Figure 6.19: Stowage Compartment Contents List for SSR 3

6.1.6 Suggestions For Future Development

Operational experience has shown that the efficiency of the data base management system could be enhanced by using a spreadsheet format instead of the information form format now implemented. For example, with a spreadsheet format, when a user selects the Job Icon, a (scrollable) spreadsheet would appear showing all the jobs and their attributes (e.g., crewmembers required, default performance time, and resource usages). Each row of the spreadsheet would correspond to a different job, and there would also be selectable icons to call up hidden information, such as job descriptions and crewmember performance times. In this way, it would be relatively easy to compare the attributes of several jobs. It would also be easy to give the spreadsheet user the capability to copy a job (or parts of a job) to make a new job with very similar attributes. The present implementation requires complete retyping of all job attributes.

It would be desirable to have the capability to express groups of jobs (performances) as a unit, and when the jobs are sent to the scheduler, the appropriate precedence constraints should be automatically generated. It should also be possible to specify maximum total time durations on a performance.

Another useful capability would be to specify that a job or performance be repeated in a flight plan several times (or as many times as possible). As it is conceivable that the utility (or importance) of scheduling later performances might be less that that of earlier ones, there should therefore be some method of prioritizing the performances.

The text editor used to input crewmember and job background information and tool usage instructions has very limited capabilities. An enhanced editor would allow the user to cut, copy, and paste text, change fonts and font sizes, provide automatic scrolling, etc.

Job resource usages and flight plan resource limits have been assumed to be constants. A more sophisticated implementation would allow additional resources and time varying usage levels and limits.

In the current implementation, the number of crewmembers needed to do a job must be explicitly specified. It might instead be advantageous to instead allow the number of crewmembers to be a variable which is a function of the total number of crewmembers assigned to the flight plan in which the job is included. For example, a sleep job (involving the entire crew) might be included in many different flight plans, and it would be easier to simply specify that the job requires all the crewmembers, instead of making several different jobs to apply to flight plans with differing numbers of crewmembers.

The current practice of using an "effective volume" for each tool and stowage compartment provides only a rough estimate of whether a tool will fit in a compartment. A more robust implementation would include the physical dimensions of each tool and compartment, and could also do some modeling to determine whether a group of tools can fit together inside of a compartment.

In order to move a tool to a new compartment, a user must do so from the "Default Location and Status" window on the tool's information form. The present version of MFIVE does not allow modification of the information in the window called up from the Stowage Information Form (Figure 6.19), which shows the contents of the stowage compartment. It would be desirable to directly "pick up" a tool from one compartment and "drop it in" another.

6.2 The Scheduler

Once a group of crewmembers and jobs have been assembled using the Flight Plan Information Form (Section 6.1.3), the scheduler can be used to enable the user to select jobs for scheduling, set time and target constraints, inspect resource usage, and allow manual and automatic task scheduling. Figure 6.20 shows the scheduling worksheet that is presented to the user upon entering the scheduler.

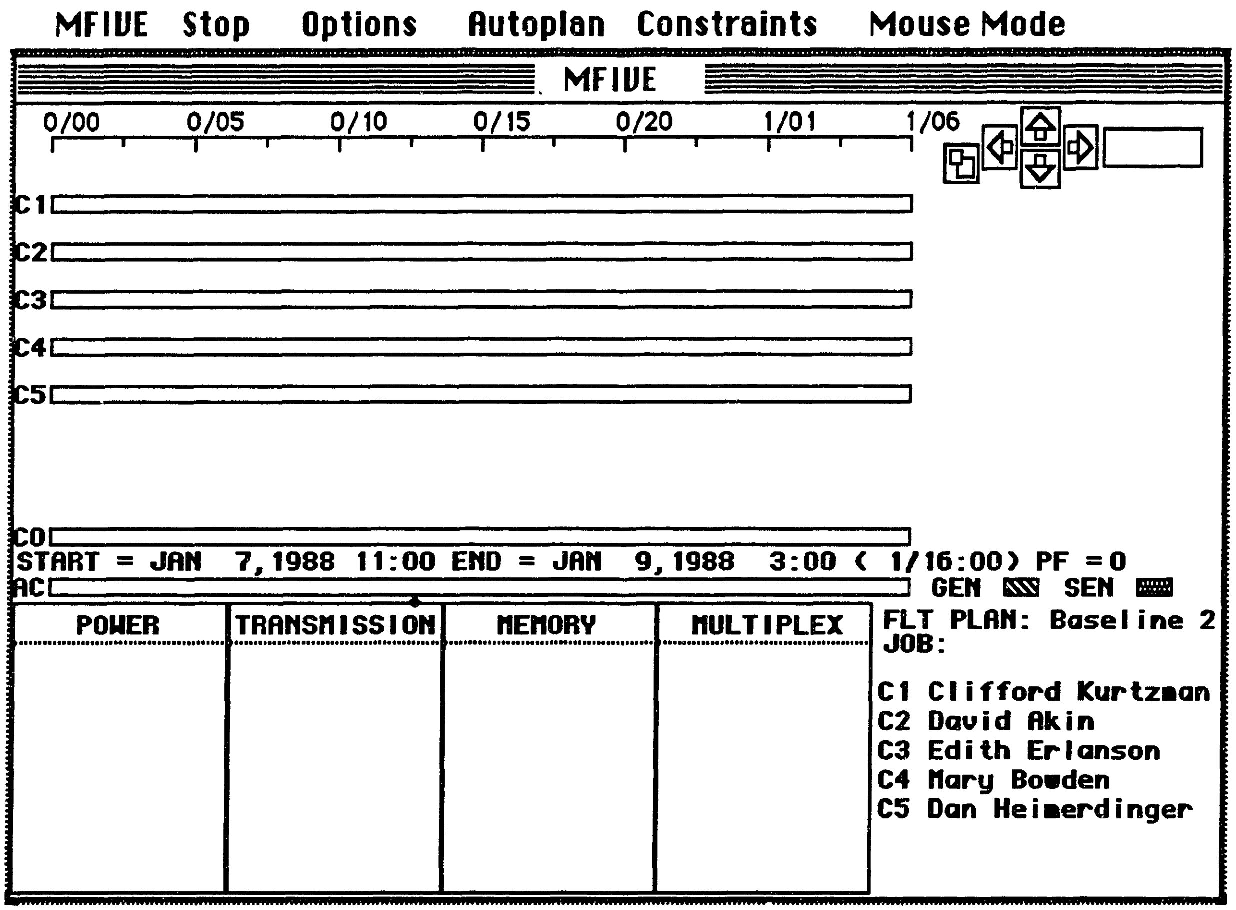


Figure 6.20: The Scheduling Worksheet

6.2.1 The Scheduling Worksheet

Central to the scheduling worksheet are the activity timelines of each of the crewmembers assigned to the flight plan. Figure 6.21 shows a flight plan midway through the scheduling process, where some of the jobs have already been scheduled. The activity timelines are the long horizontal rectangles in the upper part of the Figure 6.21. For this flight plan, there are timelines for five crewmembers, denoted C1 through C5. The names of each of these crewmembers appears in the lower right corner of the Figure 6.21.

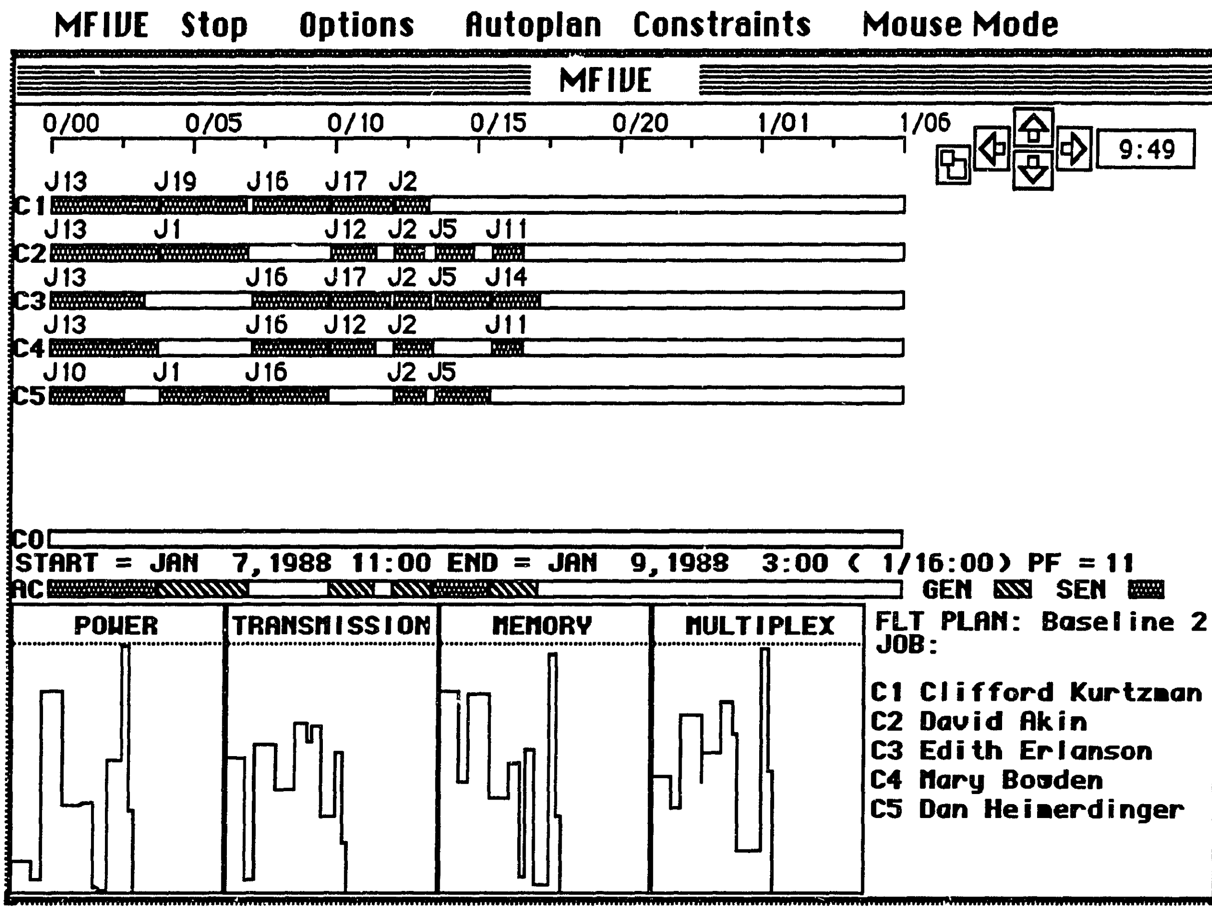


Figure 6.21: The Scheduling of a Flight Plan

Beneath the crewmembers timelines is a line of text stating the start and end dates for this flight plan. These are followed, in parentheses, by the duration of the flight plan, in the format DAYS/HOURS:MINUTES. At the top of the worksheet is a ruler showing time divisions along the timeline (in the format DAYS:HOURS). For example, the ruler in Figure 6.21 starts at zero days and zero hours (0/00) from the start of the timeline (January 7, 1988 at 11:00) and ends at one day and six hours (1/06) into the flight plan (i.e., January 8, 1988 at 17:00).

The gray shaded rectangles inside the activity timelines correspond to jobs which have been assigned to the crewmembers. For example, in Figure 6.21 we see that Job 19 (denoted J19) has been assigned to Crewmember 1 (Clifford Kurtzman) from 0/3:48 (i.e., zero days, three hours, and forty eight minutes into the flight plan) till 0/6:57. In order to find out what Job J19 corresponds to, one can click the mouse inside its shaded rectangle and a box will pop up (Figure 6.22) explaining that J19 corresponds to Job 19, "Isoelectric Focusing." In addition to the presentation of this explanation box, the exact hour and minute at which the mouse is centered is displayed in the box at the upper right corner (i.e., 5:45 in Figure 6.22). The user can thus clearly delineate the start and end times of a task by sliding the mouse from the beginning to end of the shaded rectangle.

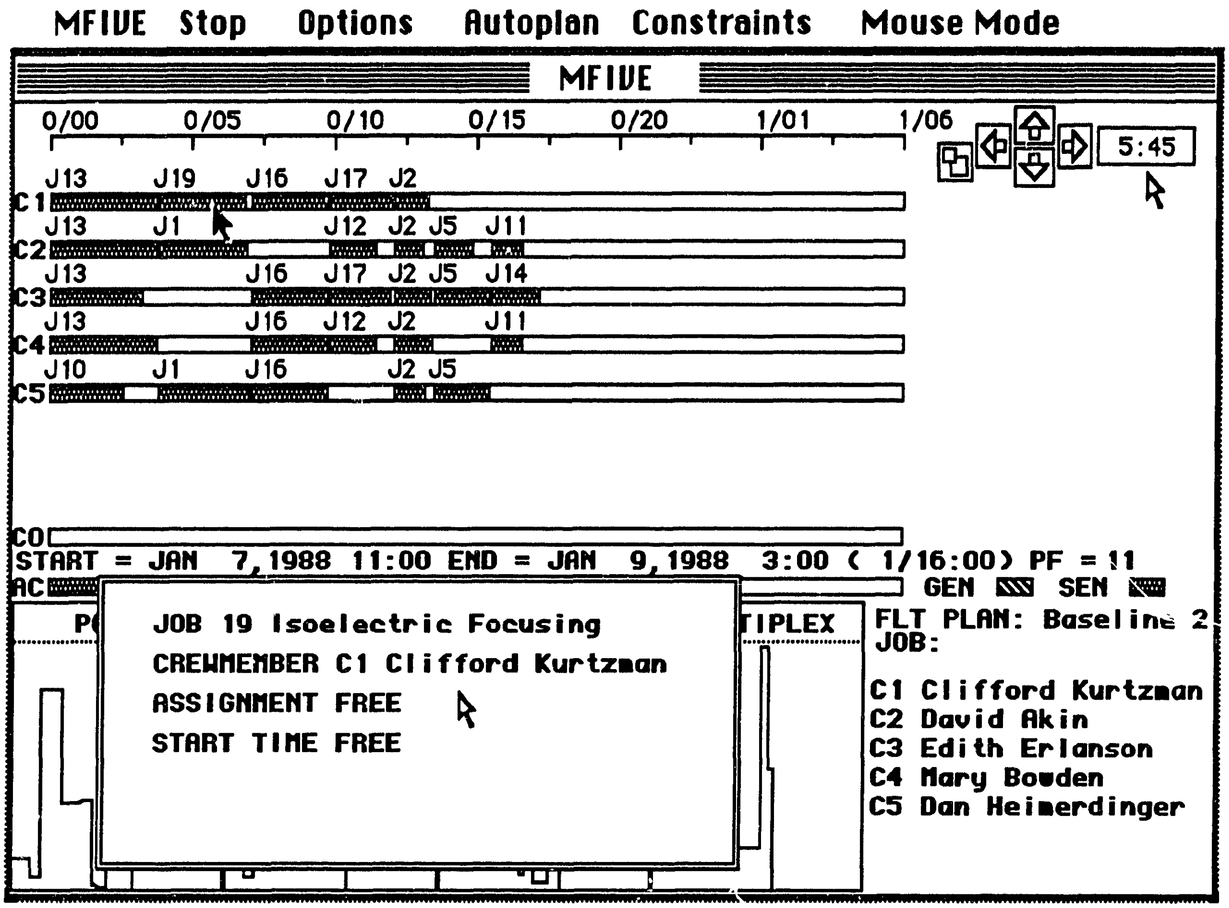


Figure 6.22: Identifying Job 19

The scale and origin of the timeline ruler at the top of the scheduling worksheet can be altered via the arrows at the upper right. In Figure 6.22, the ruler is set to start at the beginning of the flight plan, and to show the first thirty hours. This corresponds to displaying five minutes of flight plan for each pixel on the Macintosh screen. Clicking the up or down arrows either adds or subtracts one minute per pixel to the displayed timeline. For example, clicking the down arrow in Figure 6.21 results in Figure 6.23, in which the first twenty four hours of the flight plan are displayed. Note that at this four minutes per pixel setting the rectangles corresponding to the scheduled jobs are correspondingly wider.

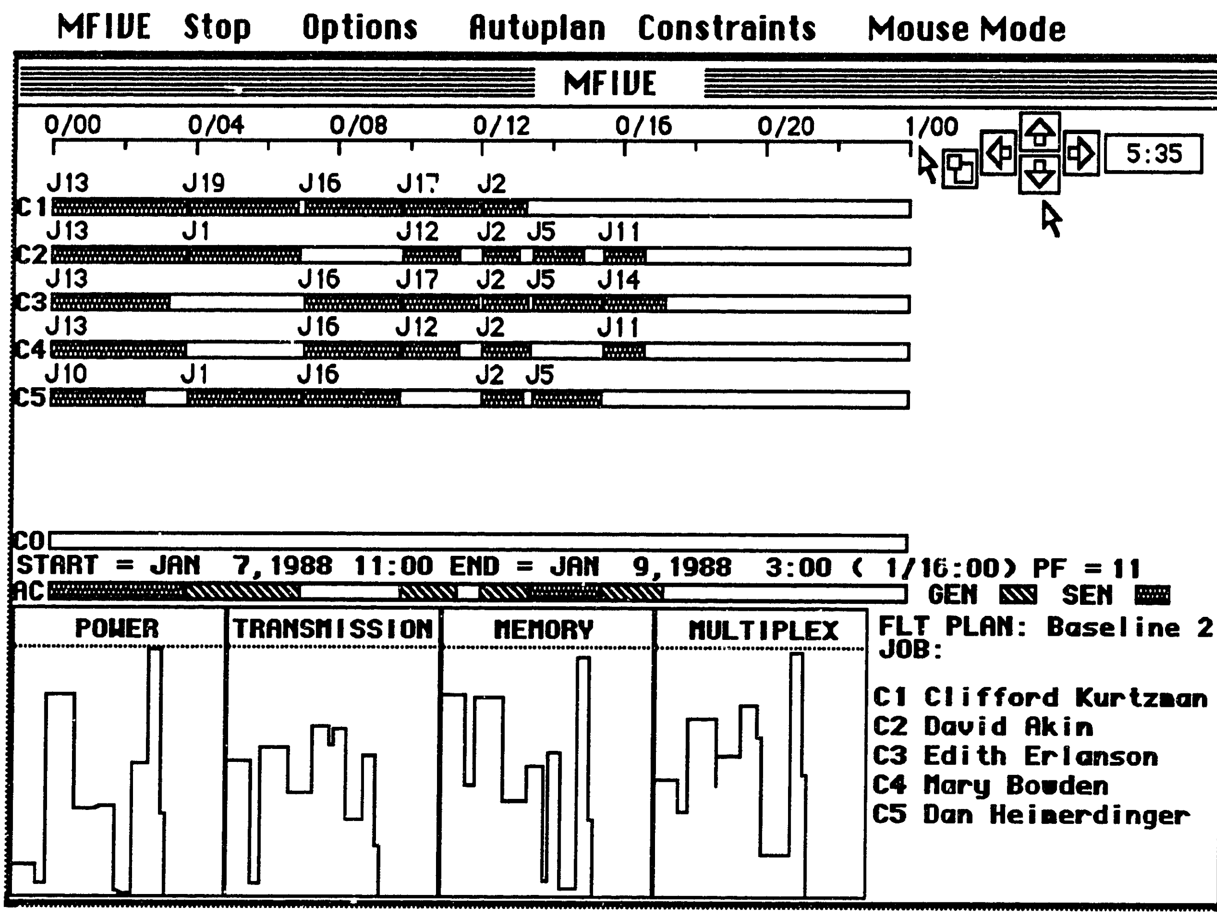


Figure 6.23: Changing the Length of the Timeline

Clicking the right or left arrow shifts the origin of the timeline. For example, clicking the right arrow in Figure 6.21 results in Figure 6.24, in which the timelines start at 0/10 and end at 1/16. In general, these arrows shift the timeline over by two thirds of its current width (i.e., 2/3 x 6 hours = 4 hours: therefore the timeiine was shifted over 4 hours).

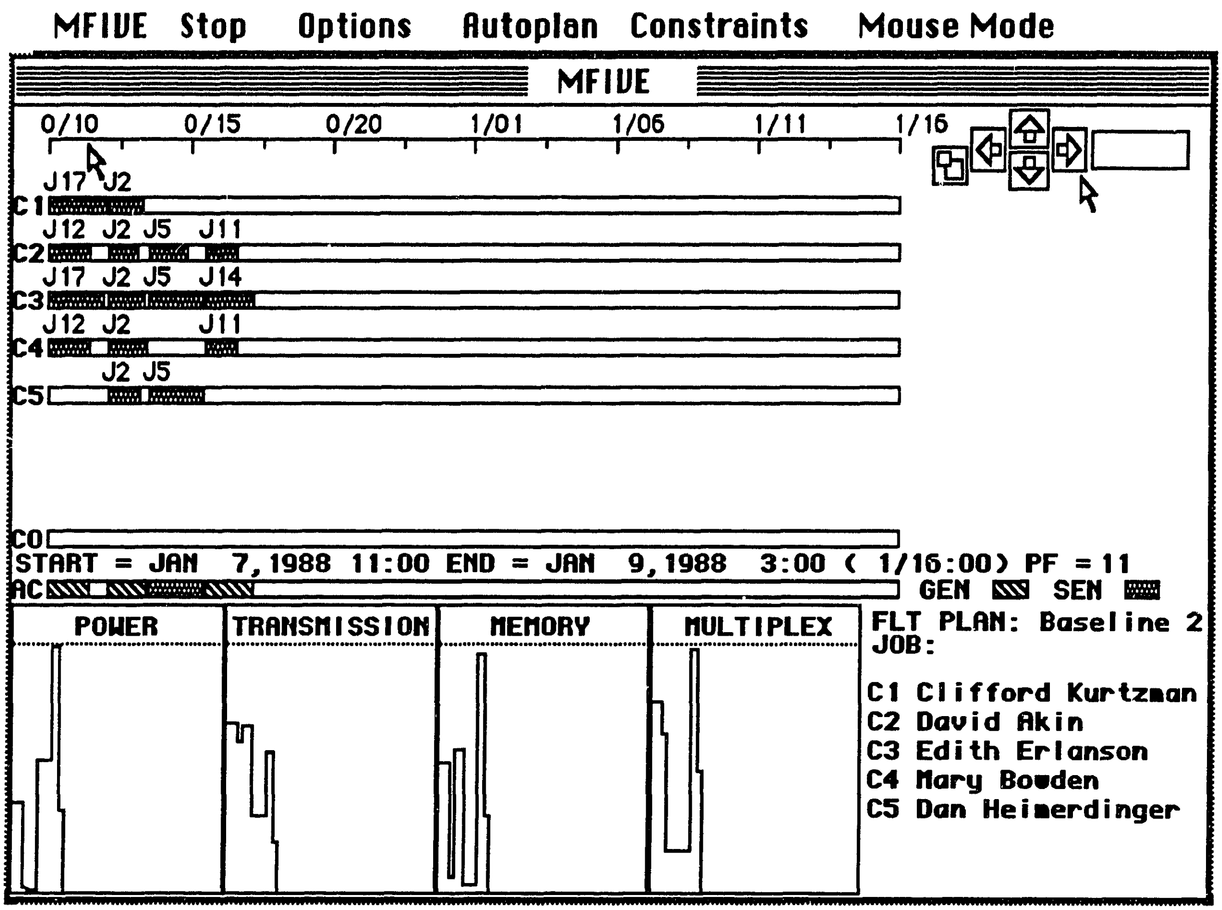


Figure 6.24: Shifting the Timeline

Clicking on the box with the two rectangles inside of it (next to the arrows) presents the user with a window in which the number of minutes per pixel and the origin of the timeline can be explicitly set (Figure 6.25).

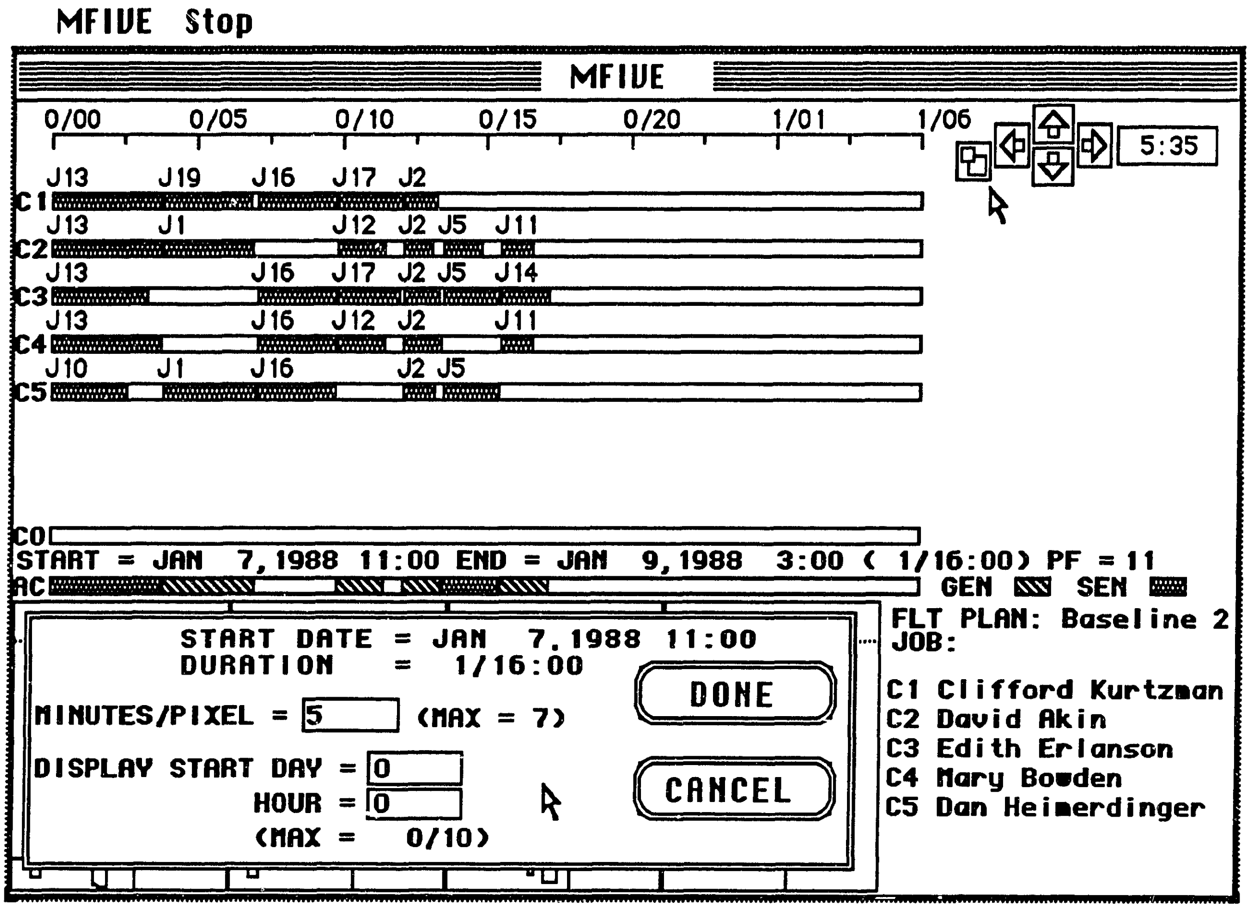


Figure 6.25: Manually Changing Timeline Origin and Resolution

The four large boxes at the bottom of the scheduling worksheet display the resource usage for the timeline. The width that each of these boxes displays corresponds to the width of the scheduling rectangles (i.e., from 0/00 till 1/06 in Figure 6.21). Changing the width or origin of the ruler changes the resource displays as well (e.g., see Figure 6.23 and Figure 6.24). The height of the resource boxes are scaled so that the dashed line near the top indicates usage at the maximum specified for the flight plan. By clicking on a resource box, the user can call up a window displaying the usage in expanded form, and with the axes labelled (Figure 6.26). The two rectangles at the bottom of the window display an exact digital reading of the resource level at the time indicated (by pointing and clicking on the mouse).

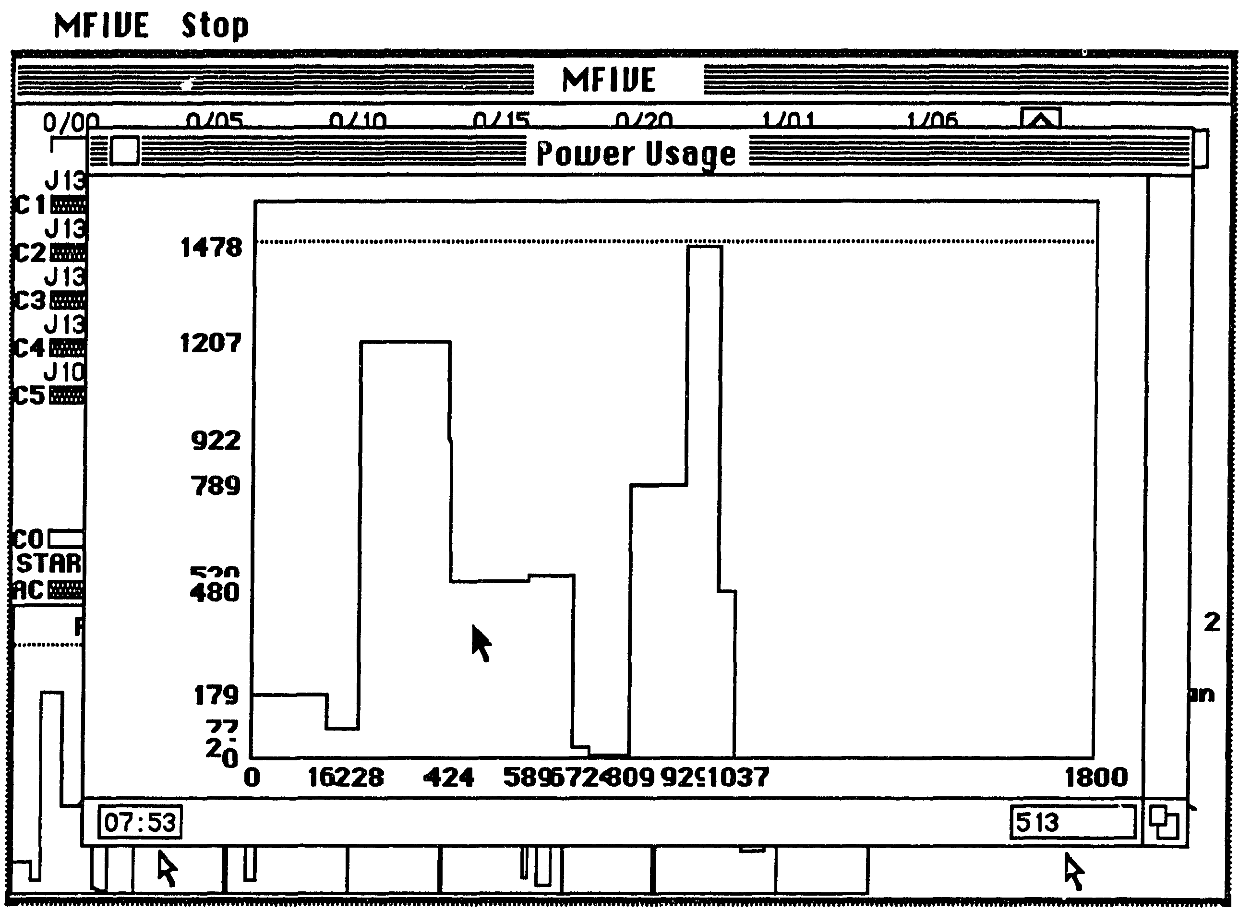


Figure 6.26: Expansion of Display of Power Usage Level

Directly above the four resource boxes, is a horizontal rectangle (labelled AC) which displays the audio constraint. Times at which a noise generating activity is being performed are signified by striped rectangles, and times at which a noise sensitive activity is being preformed are signified by gray rectangles. In Figure 6.21 it can be seen that there are noise generating rectangles corresponding with the durations of Jobs 1, 19, 12, 2, and 14.

6.2.2 Selecting a Job to be Scheduled

There are three ways to select a job for scheduling. Figure 6.27 shows the OPTIONS menu, at the top of the Macintosh screen. The user can manually determine the next job to be scheduled by selecting the option DISPLAY JOB INFO. This will present the information window shown in Figure 6.28. This window contains information relating the number of crewmembers required, the default time, the minimum time for any crewmember, the resource usages, and the audio status of each job (some of this information is not visible in the portion of the window shown in Figure 6.28.). An asterisk next to the name of a job indicates that it has already been scheduled.

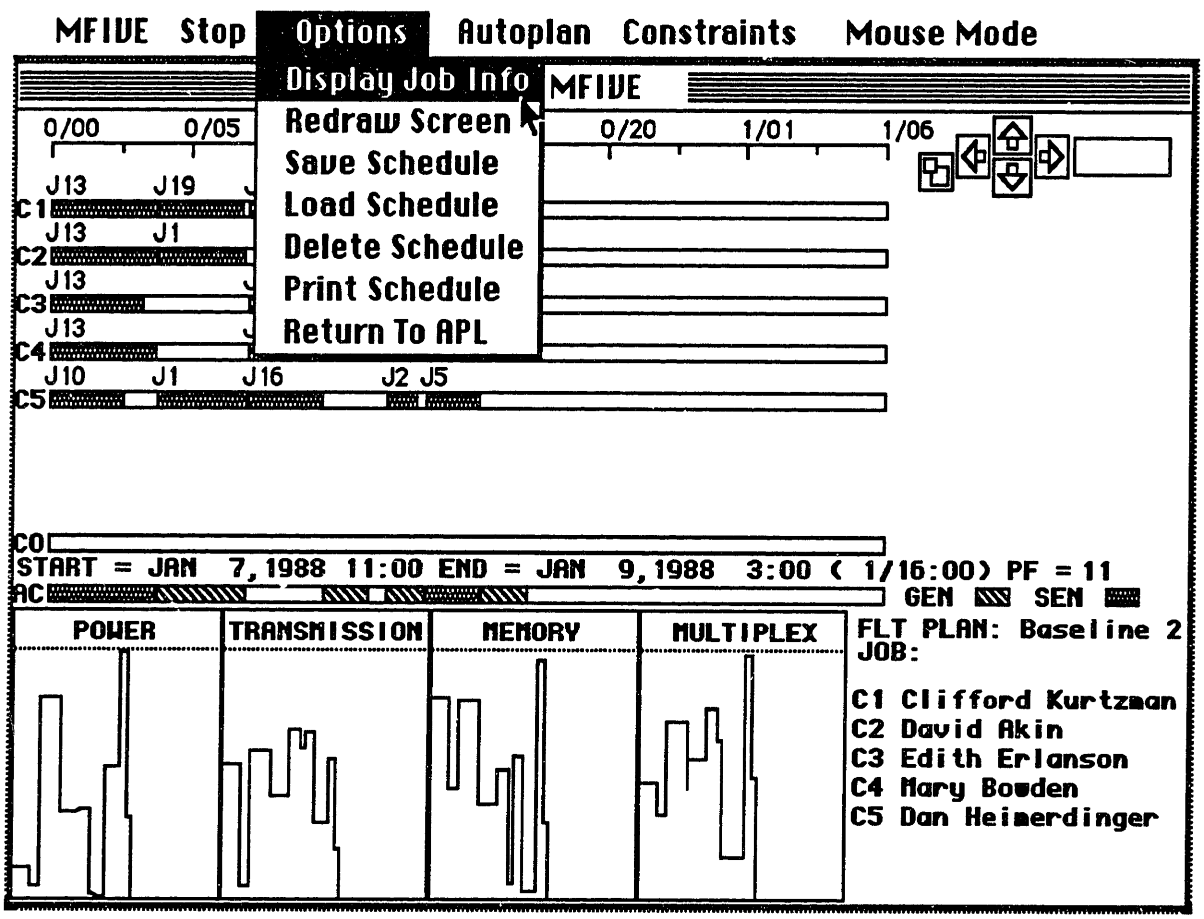


Figure 6.27: Selecting SELECT JOB INFO from the OPTIONS Menu

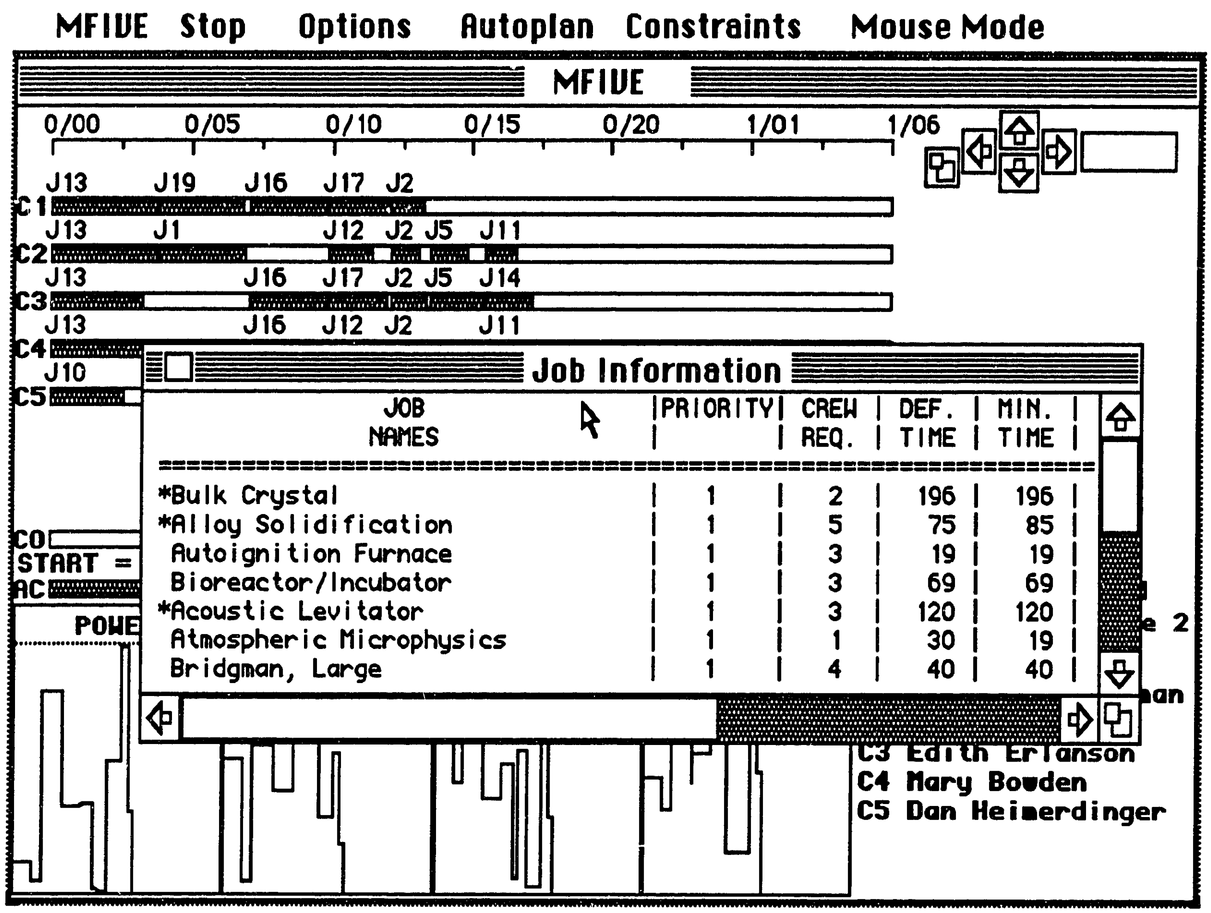


Figure 6.28: The Job Information Window

Clicking on the name of a job selects that job for scheduling. Figure 6.29 shows how the scheduling worksheet looks after Continuous Flow Electrophoresis (Job 9) has been selected, and the job information window has been closed. Immediately above the names of the crewmembers appears the name of the job being scheduled, and below the right arrow at the top appears the number of crewmembers necessary to perform this job.

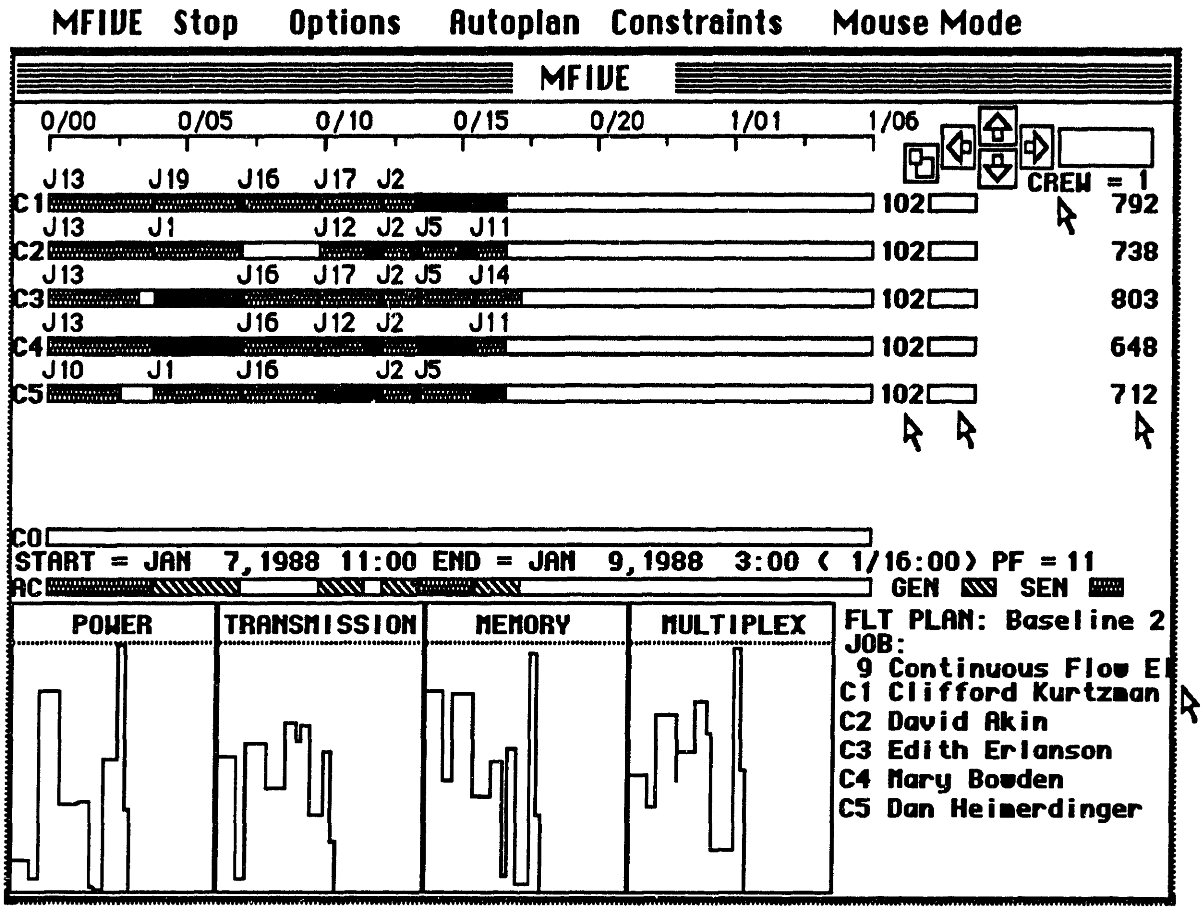


Figure 6.29: Selecting Job 9 For Scheduling

Immediately to the right of the activity timelines appears the number of minutes it takes each of the crewmembers to perform this job. For example, Figure 6.29 shows that each of the crewmembers takes 102 minutes to perform this job. Next to the numeric performance times are rectangles which have a width corresponding to the crewmembers performance time. To the right of the rectangles are numbers indicating the total workload (in minutes) for which each of the crewmember have already been scheduled. For example, in Figure 6.29 Crewmember 2 has been assigned a total of 738 minutes of worktime.

When a job has been selected for scheduling, parts of the crewmembers activity timelines may become "blacked out", as in Figure 6.29. This is because these time slots are infeasible due to one or more constraint violations. To find the reason an interval is blacked out, the user can simply click the mouse on the blackened region, and a window will appear with an explanation (Figure 6.30). In the case of Figure 6.29, the blackened intervals are due to the fact that scheduling Job 9 in these intervals would exceed both the power and data transmission resource limits.

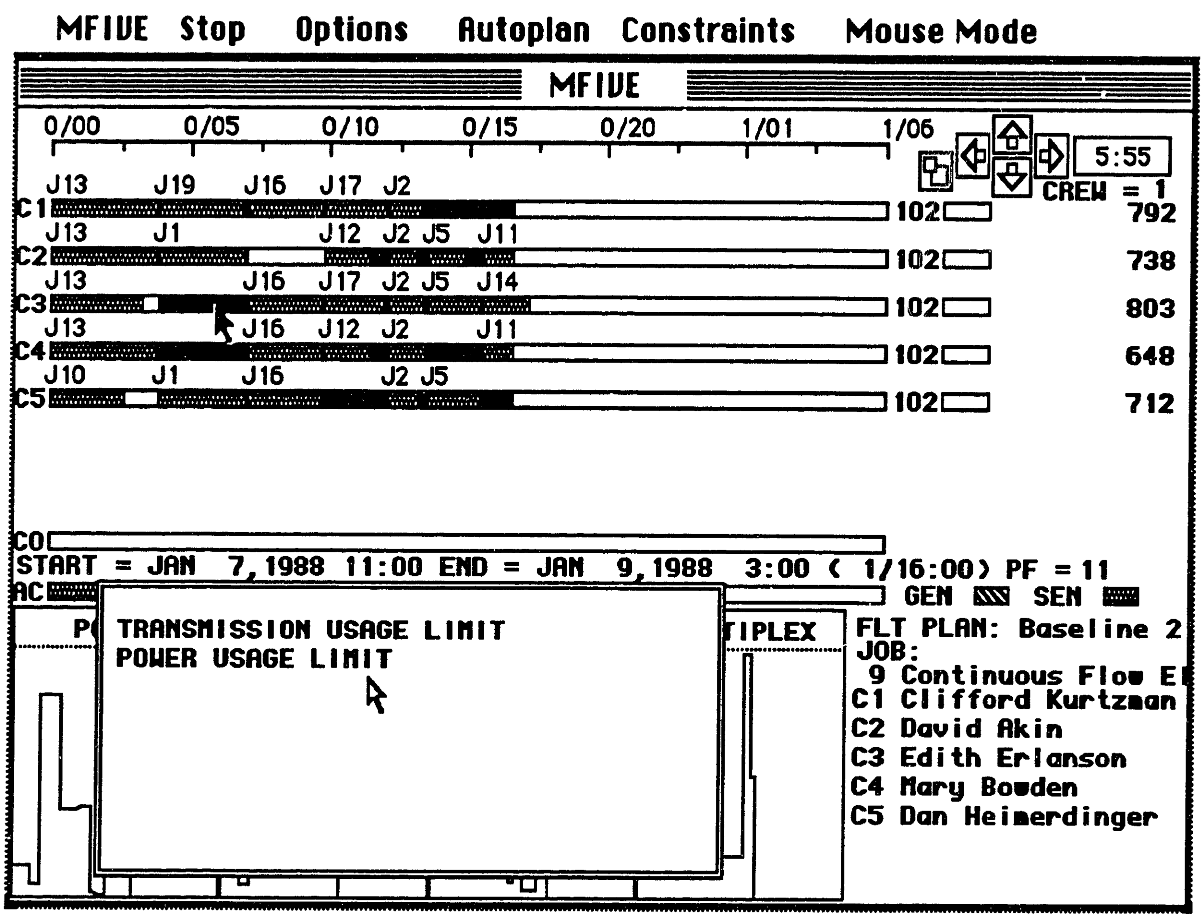


Figure 6.30: Clicking on a Blacked Out Region for an Explanation of a Constraint Violation

The second method of selecting a job for scheduling is to choose the SELECT JOB option from the AUTOPLAN menu (Figure 6.31). This will instruct MFIVE to pick the next job to be scheduled, based upon its internal heuristic (see Section 5.1.1). The user can select the heuristic used by MFIVE through the SET HEURISTIC option, also on the AUTOPLAN menu (Figure 6.32). Choosing the AUTO JOB SELECT option has the same effect as the SELECT JOB option, except that after the job which MFIVE selects is scheduled, MFIVE will automatically pick another job for scheduling.

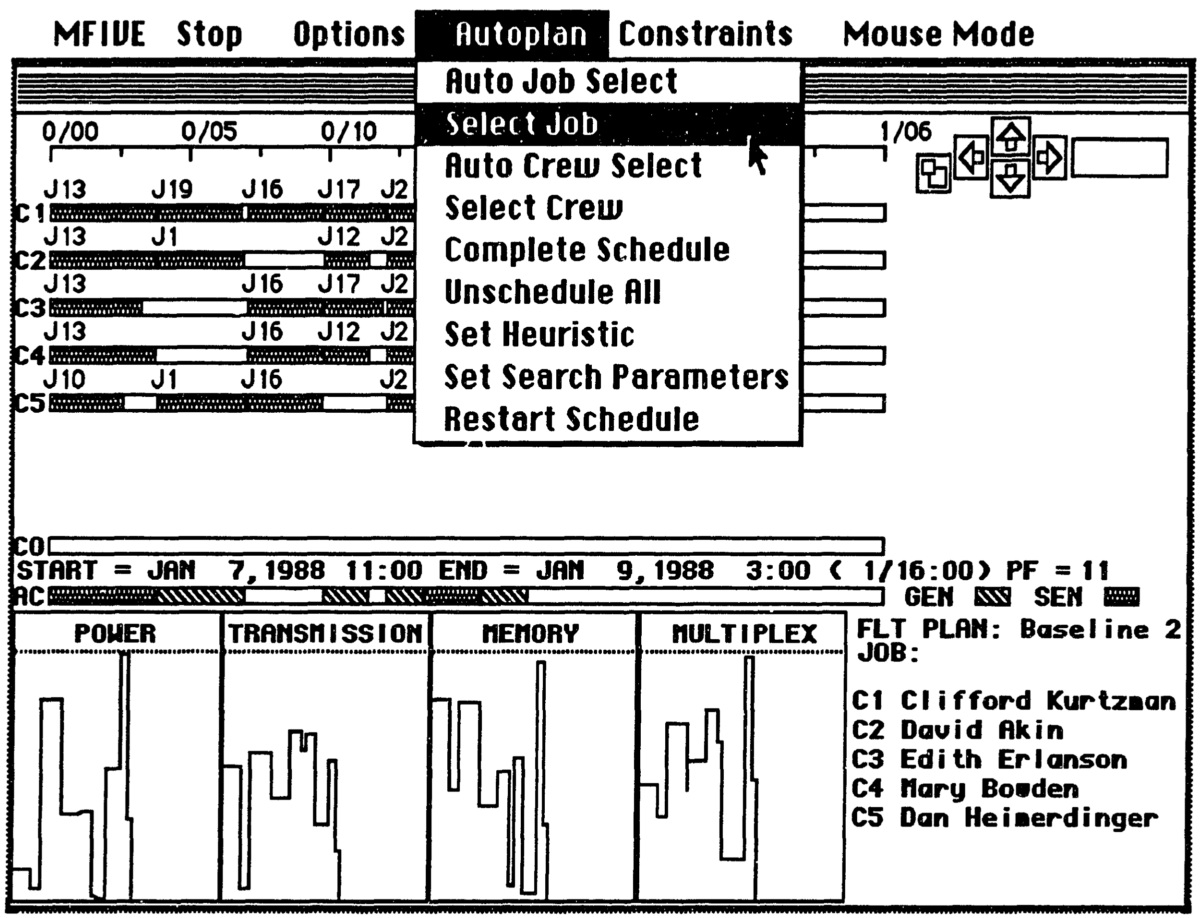


Figure 6.31: Choosing "Select Job" from the AUTOPLAN Menu

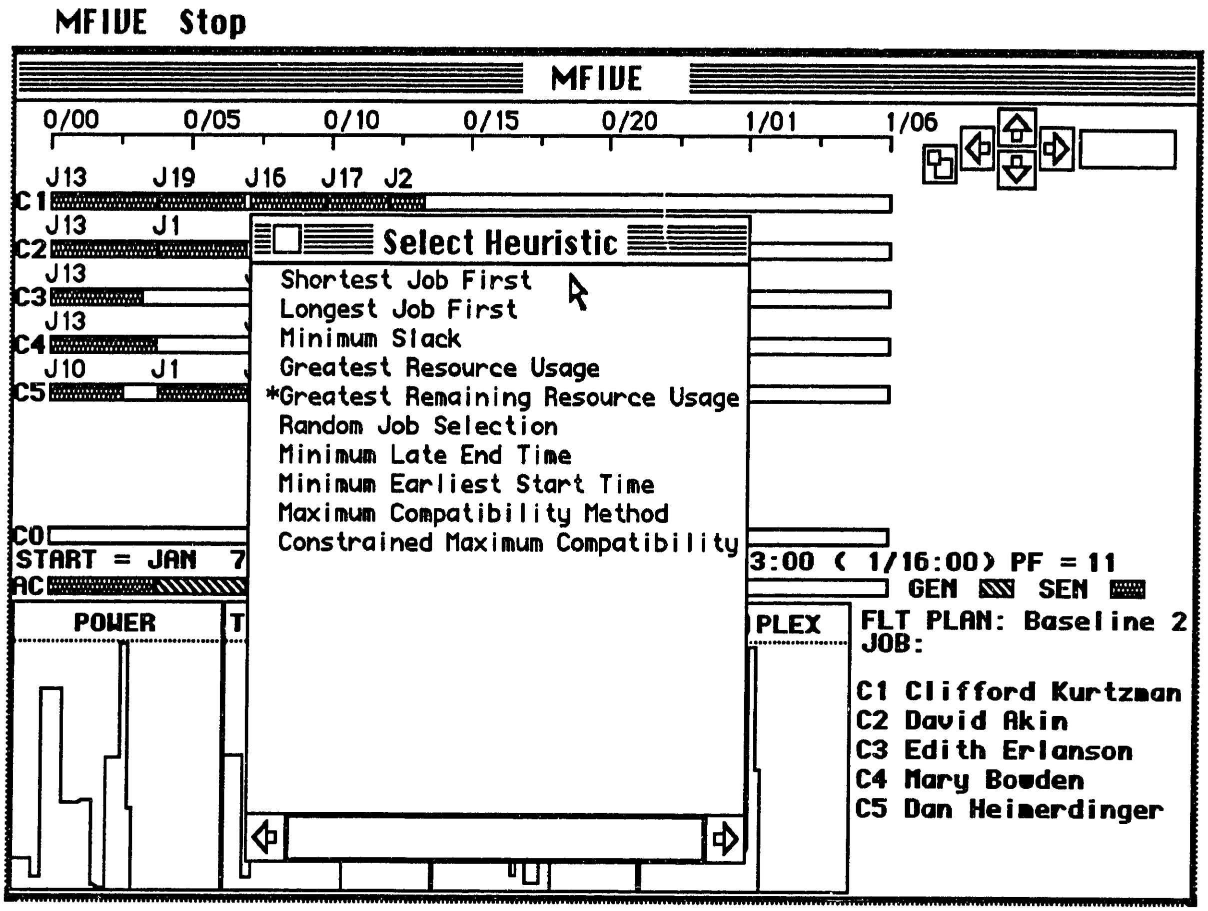


Figure 6.32: Selecting an Heuristic

6.2.3 Jobs Requiring a Single Crewmember

Once a job has been selected for scheduling, a crewmember may be assigned by several methods. The user can click on the crewmember's rectangle (to the right of the activity timeline) and slide this rectangle (with the mouse) to the desired spot on the crewmembers timeline. The exact time at which the rectangle is positioned is always displayed at the upper right of the screen, so that the rectangle can be accurately positioned. Figure 6.33 shows the result of sliding Crewmember 5's rectangle to the 18:35 position, and then releasing the mouse. The resource usages are appropriately updated.

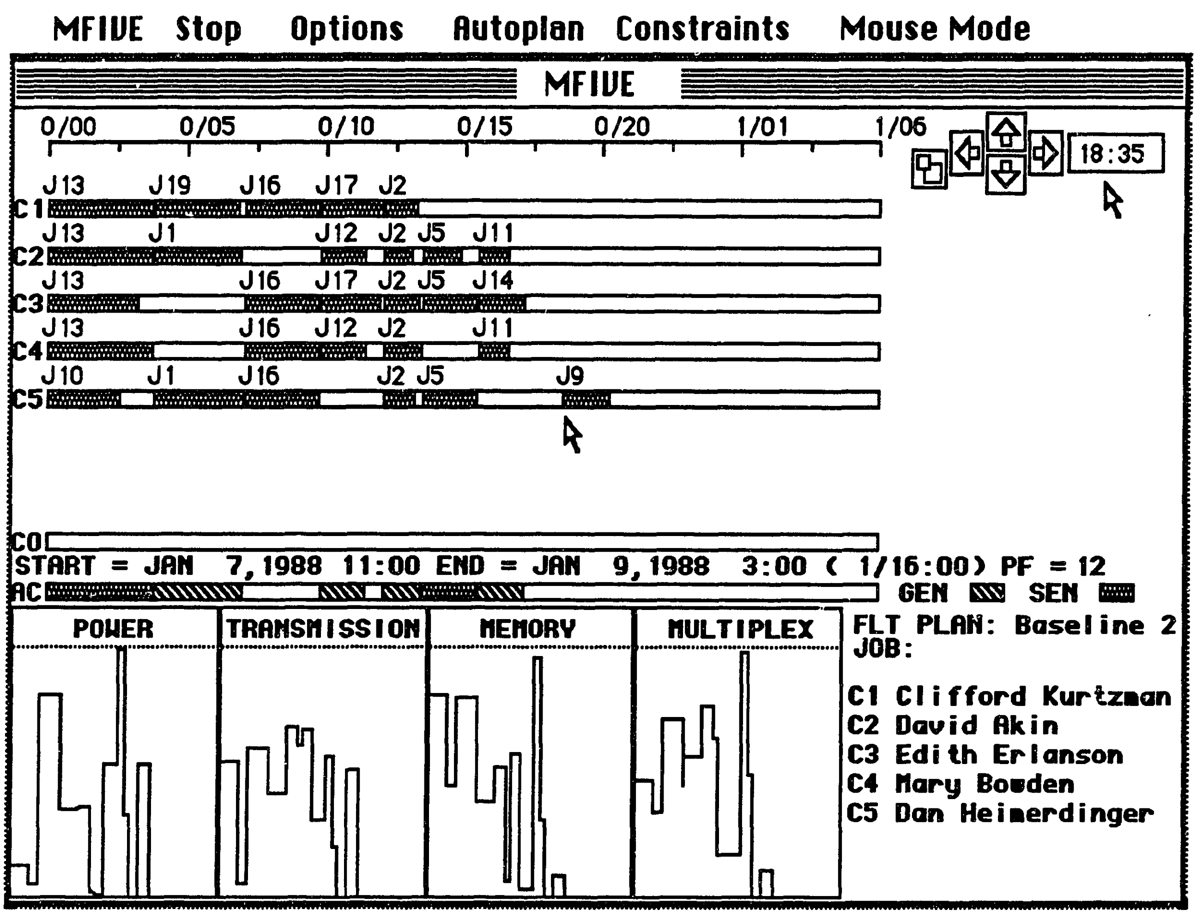


Figure 6.33: Manually Scheduling Job 9

In order to assign a job to a crewmember at the crewmember's earliest available start time, the user can click the mouse on the crewmember's number (e.g., C5, at the very left of Crewmember 5's timeline). Figure 6.34 shows the result of clicking on C5.

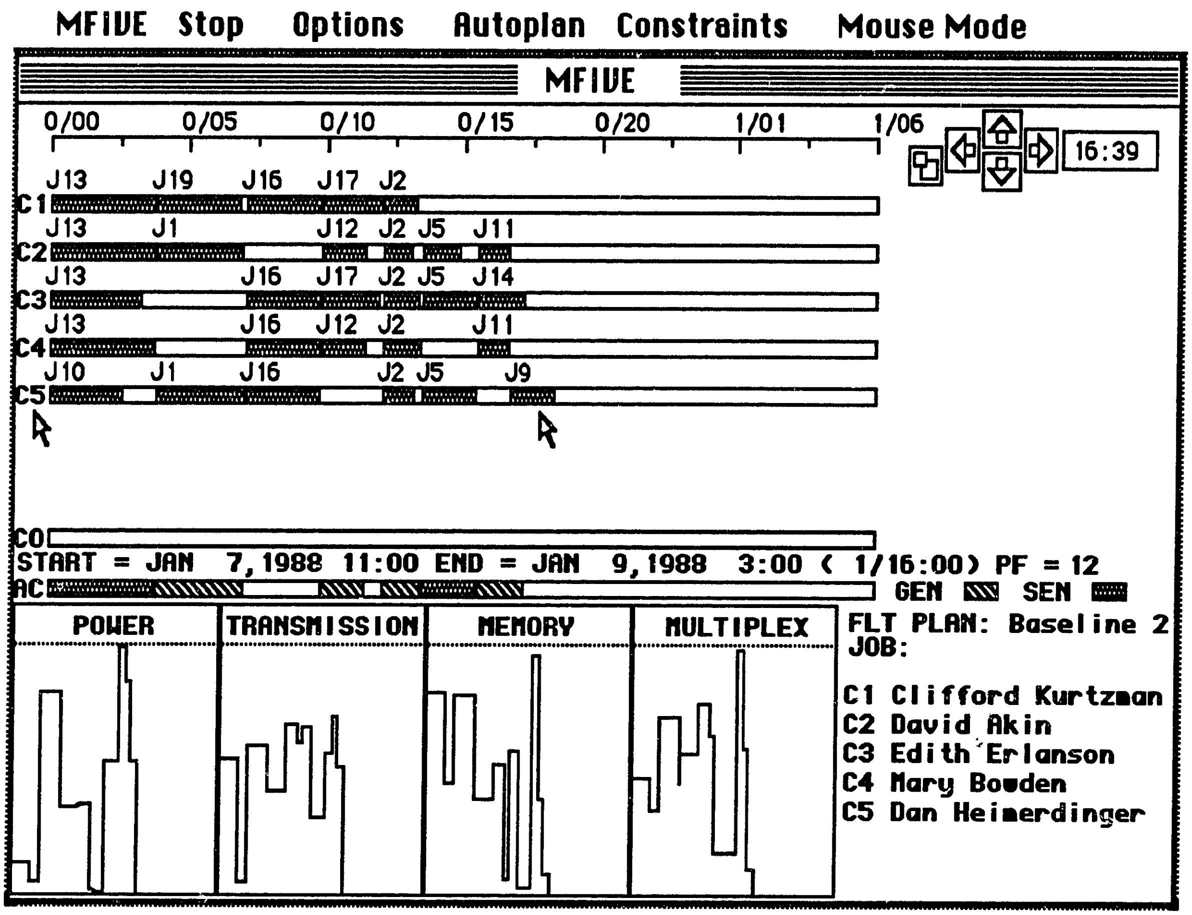


Figure 6.34: Scheduling Job 9 at Crewmember 5's Earliest Feasible Time

Crewmember selection can be completely automated by choosing the SELECT CREW option from the AUTOPLAN menu. MFIVE will then choose the best crewmember based upon its internal heuristic (see Section 5.1.2), and assign the job to that crewmember at the crewmember's earliest available time (Figure 6.35). In this case the internal heuristic assigns the task to Crewmember 2.

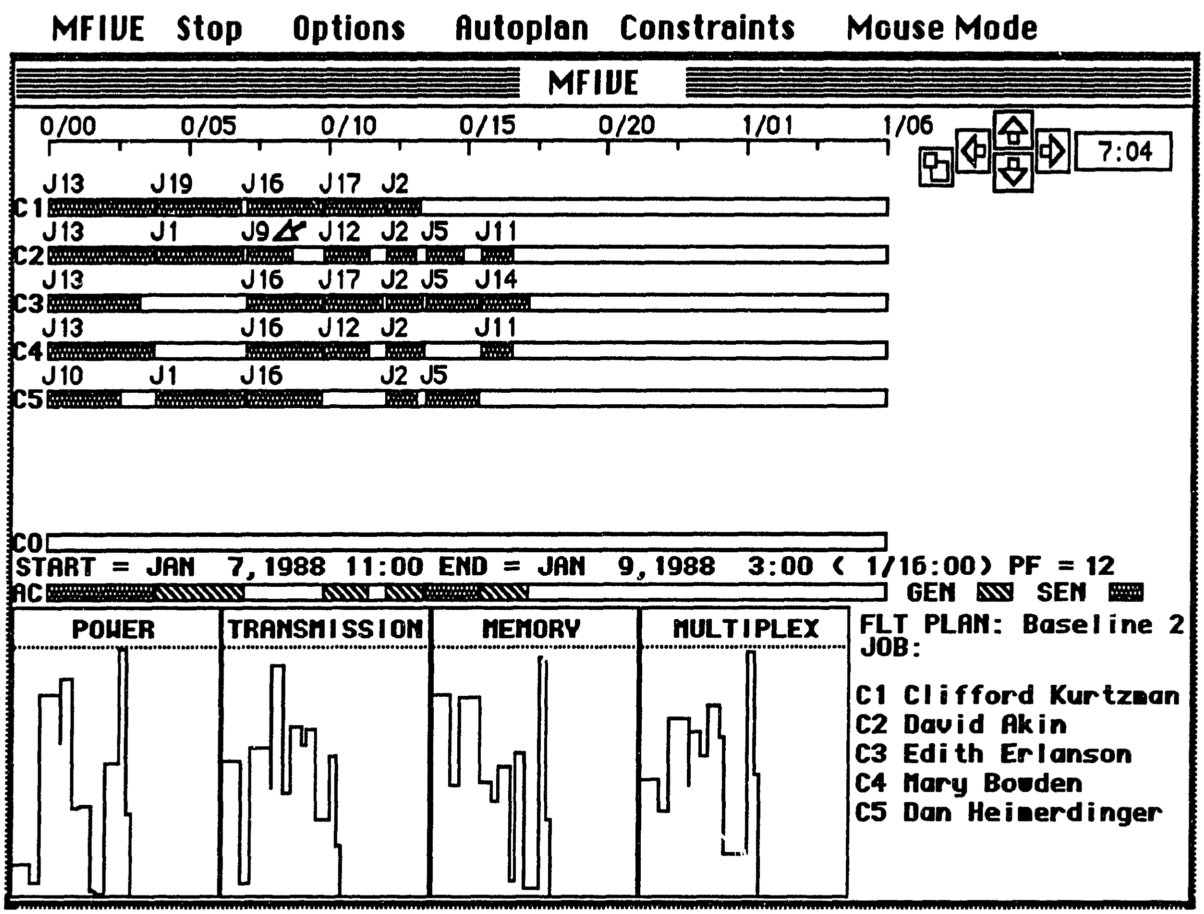


Figure 6.35: Scheduling Job 9 Using SELECT CREW

Choosing the AUTO CREW SELECT option from the AUTOPLAN menu will have the same effect as the SELECT CREW option, but then after every subsequent job is selected for scheduling, the crewmember will automatically be assigned. By selecting both the AUTO CREW SELECT option and the AUTO JOB SELECT option, MFIVE will autonomously attempt to plan out the entire remaining flight plan. If a situation is reached where some job cannot possibly be scheduled, it will be skipped and another job will be selected for scheduling.

6.2.4 Jobs Requiring Multiple Crewmembers

Jobs requiring multiple crewmembers can be scheduled in a manner similar to that for single crewmember jobs. For manual scheduling, after the user has slid into place the rectangle for one of the crewmembers, MFIVE will generate striped rectangles for each of the other crewmembers which are eligible to perform the job at the same start time (Figure 6.36). The user then clicks on these rectangles until the required number are selected. (If the required number is exactly equal to the number of crewmembers available, then MFIVE will automatically perform the selection.) Resource usage is taken to extend over the period covered by the crewmember taking the longest to perform the job.

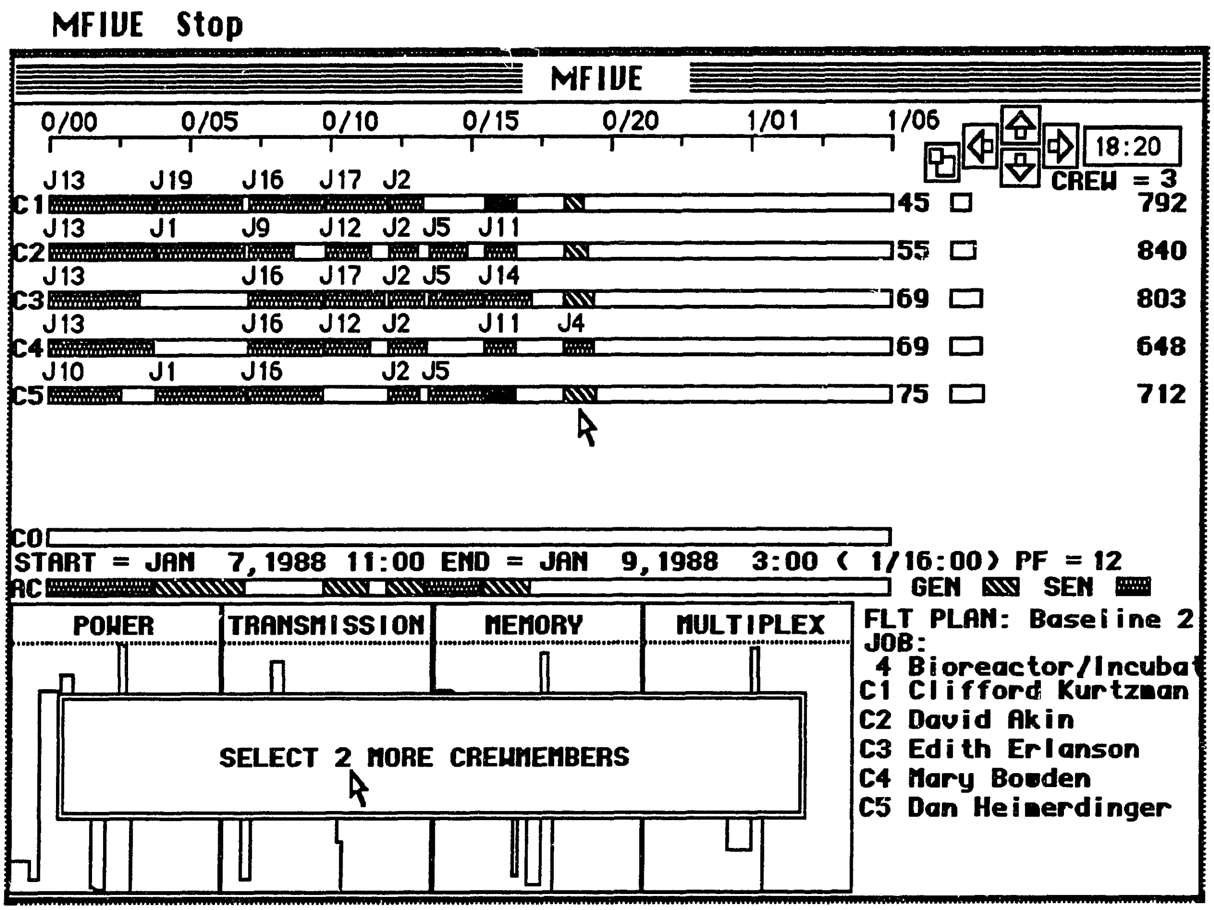


Figure 6.36: Manually Scheduling a Job Requiring 3 Crewmembers

The job can be assigned to a group of crewmembers at their mutually earliest start time by successively clicking on the numbers of those crewmembers. Alternatively, SELECT CREW and AUTO CREW SELECT options can also be used to allow MFIVE to autonomously choose crewmembers and time slots for the task.

6.2.5 Jobs Requiring No Crewmembers

MFIVE is capable of scheduling jobs which do not require any crewmembers. For example, during a materials processing experiment, one step (job) might require that a sample be heated for several hours, without any crewmember supervision necessary. The scheduling bar, labelled **C0** on the scheduling worksheet, is for scheduling and displaying these jobs. If such a job is selected for scheduling, a rectangle whose width corresponds to the length of the job will appear to the right of the scheduling bar. The job can then be scheduled by manually moving this rectangle to the desired start time, by clicking on **C0**, or by using the SELECT CREW or AUTO CREW SELECT option. It should be noted that while it is impossible for a crewmember to do more than one job at any given time, it is possible for several jobs which do not require any crewmembers to occur simultaneously. If the user clicks on a grey rectangle on the **C0** scheduling bar, a window will pop up displaying the names of all the jobs occurring at that time.

6.2.6 Schedule Improvement by Iteration

By selecting the COMPLETE SCHEDULE option from the AUTOPLAN menu, the user can begin an iterative search for an optimized schedule (see Section 5.2.4). The parameters for the search are set by selecting the SET SEARCH PARAMETERS menu on the AUTOPLAN menu (Figure 6.37). This will allow the user to specify whether or not the sampling method (randomization of priorities) is used for job ordering (Section 5.2.4.2). It is also possible to specify limits on the amount of time allowed for the search, and to designate the optimality criteria used for determining better schedules. The scheduler always tries to maximize the sum of the priorities of the jobs which are successfully scheduled (see Section 6.2.5). However, among schedules with equal priorities (such as all schedules which are successful in scheduling all the jobs), the user can either set the scheduler to minimize the completion time of the final job in the schedule, or to minimize the average completion time of the jobs.

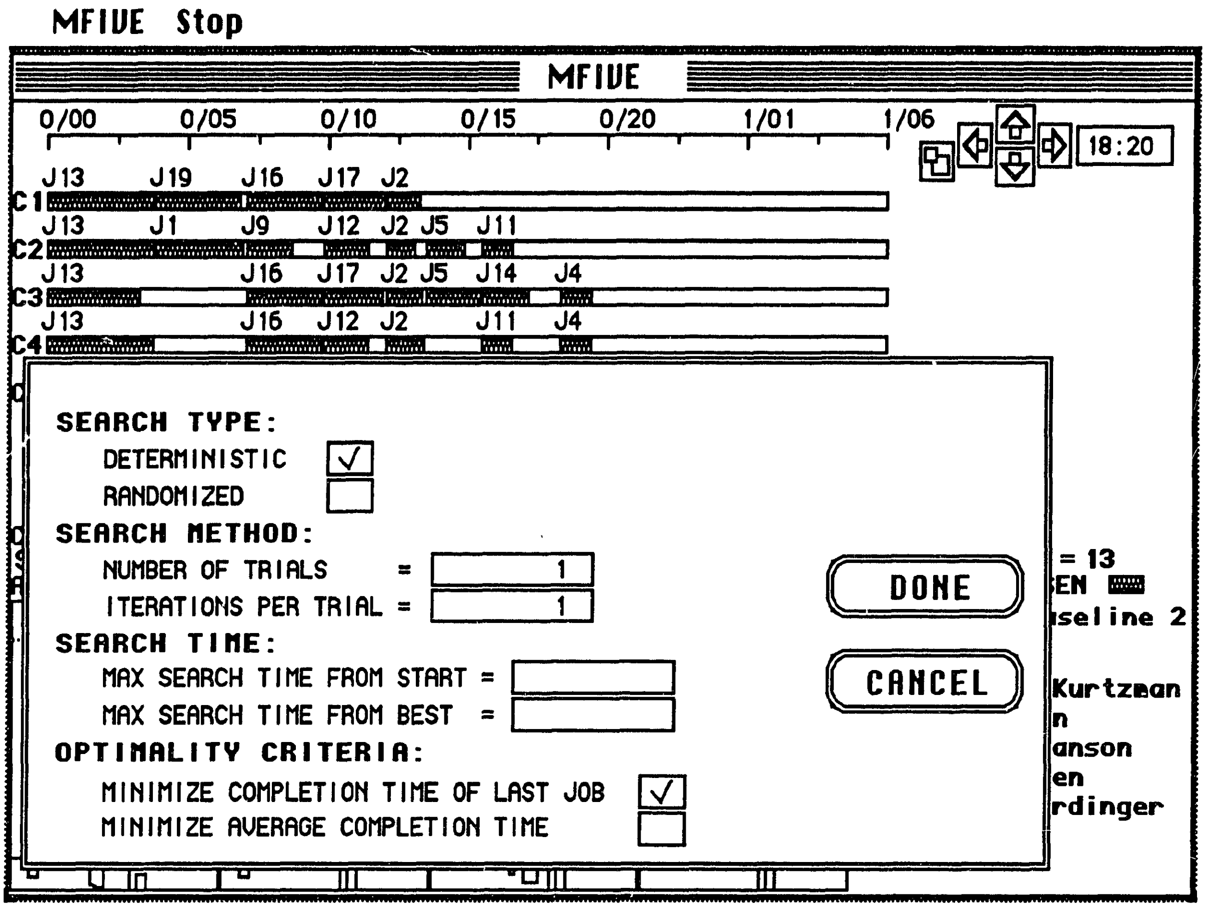


Figure 6.37: Setting the Search Parameters

The SET PARAMETERS form also allows the user to specify the number of trials and the number of iterations per trial. Multiple trials of the same algorithm may yield different results because the algorithms employ some degree of randomness in moving from iteration to iteration, as discussed in Section 5.2.4.1. At the completion of the search, MFIVE will present the user with the best solution found from among all the trials attempted.

6.2.7 Job Priorities

The CONSTRAINTS menu, Figure 6.38, allows the user to define job priorities, and add time constraints (such as earliest start time and precedence constraints) and target constraints.

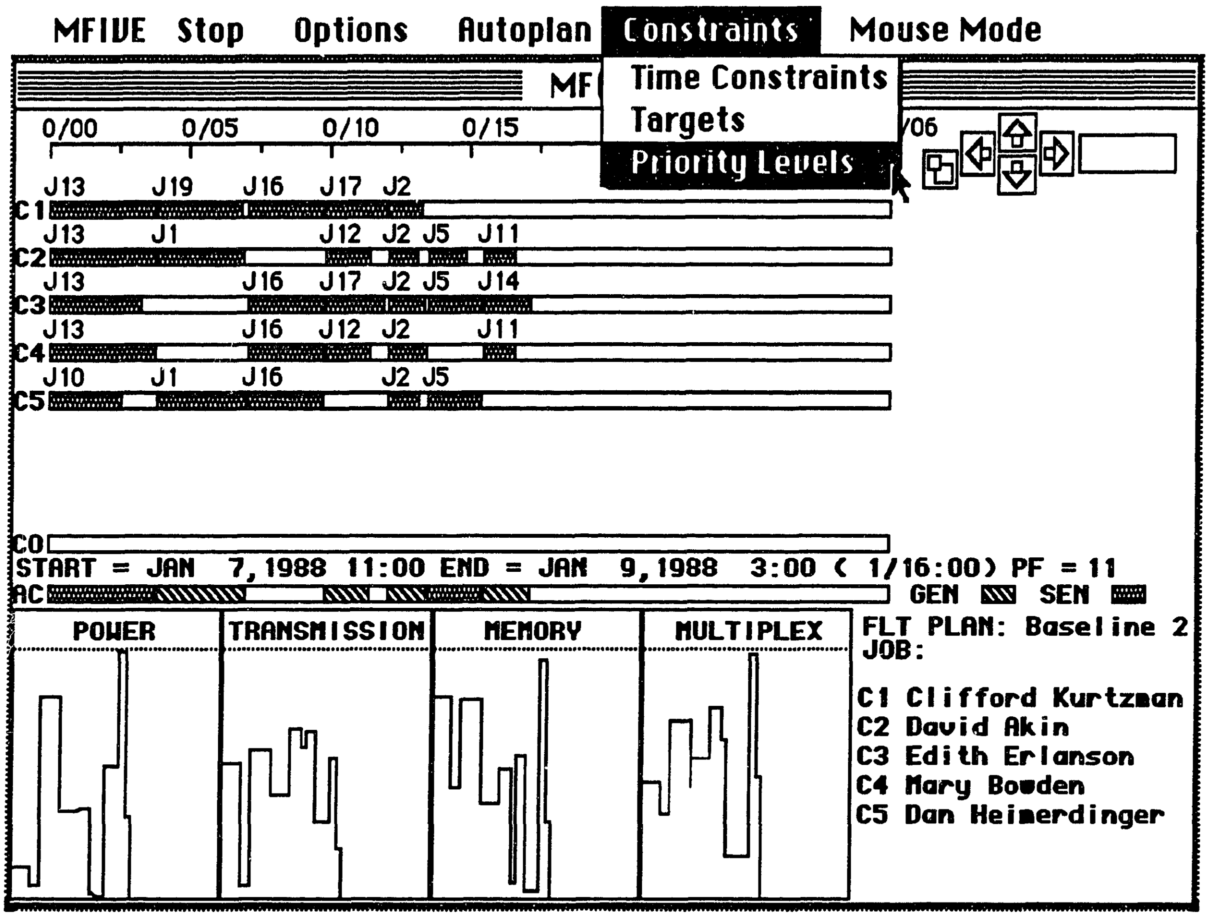


Figure 6.38: Selecting PRIORITY LEVELS from the CONSTRAINTS Menu

Selecting PRIORITY LEVELS from the CONSTRAINTS menu gives the user a window (Figure 6.39) containing the relative priorities of each of the jobs. Initially, each job has a priority of 1, but the user can replace this with a higher value to indicate an increased importance for the job, or with a lower value, to indicate a decreased importance. The sum of the priorities of the jobs scheduled (in a partial schedule) is indicated following the letters "PF =" on the same line as the start and end dates of the flight plan.

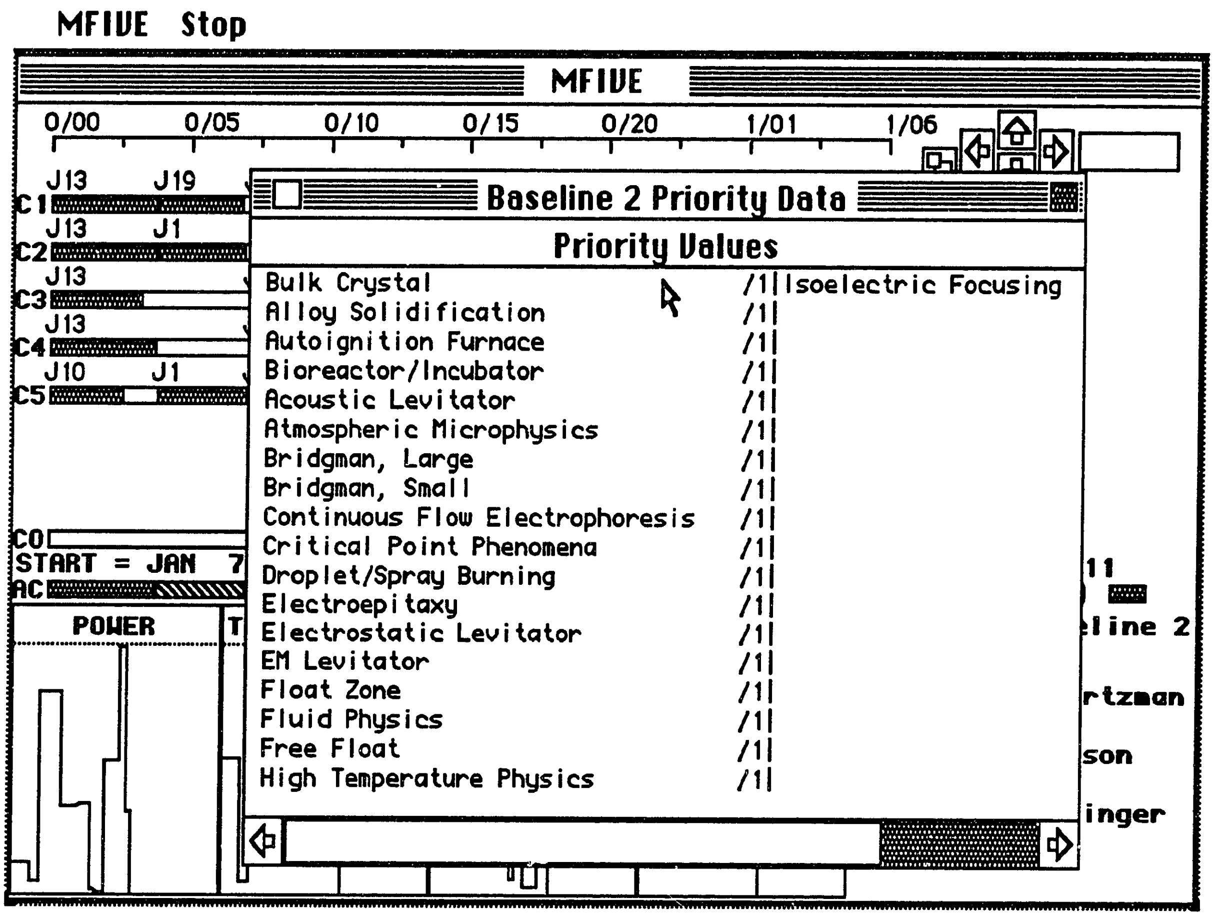


Figure 6.39: The Job Priorities Window

6.2.8 Time Constraints

Choosing the TIME CONSTRAINTS option from the CONSTRAINTS menu gives the user a window in which time constraints can be added and modified (Figure 6.40). Listed in the first four columns of this window are the names of the jobs, their earliest start times (EST), latest start times (LST), and latest end times (LET). For example, in Figure 6.40 it can be seen that for the job BULK CRYSTAL, its earliest start time is at the start of the flight plan (i.e., zero days, zero hours and zero minutes). For this same job, the latest start time is at 11 hours and 17 minutes into the flight plan, and the latest end time is at exactly 15 hours into the flight plan.

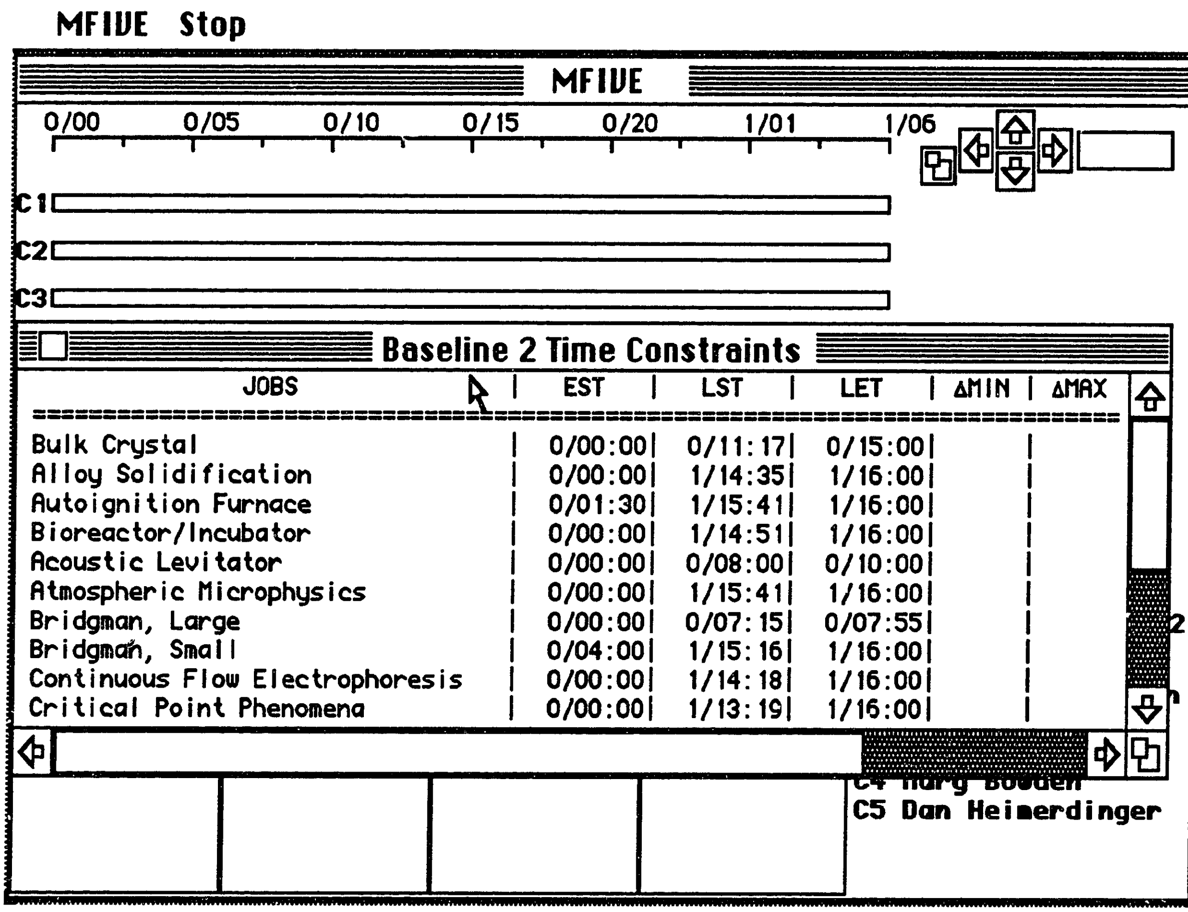


Figure 6.40: The Time Constraints Window

Data can be revised by clicking on an item. For example, in Figure 6.41, the latest end time for ALLOY SOLIDIFICATION has been selected for revision. Here a new constraint is added, m:.king the latest end time of this job to be 0 days, 12 hours, and 25 minutes into the flight plan. In Figure 6.42, the user has clicked on the job ACOUSTIC LEVITATOR (note also that the previous revision to ALLOY SOLIDIFICATION has been entered into the table).

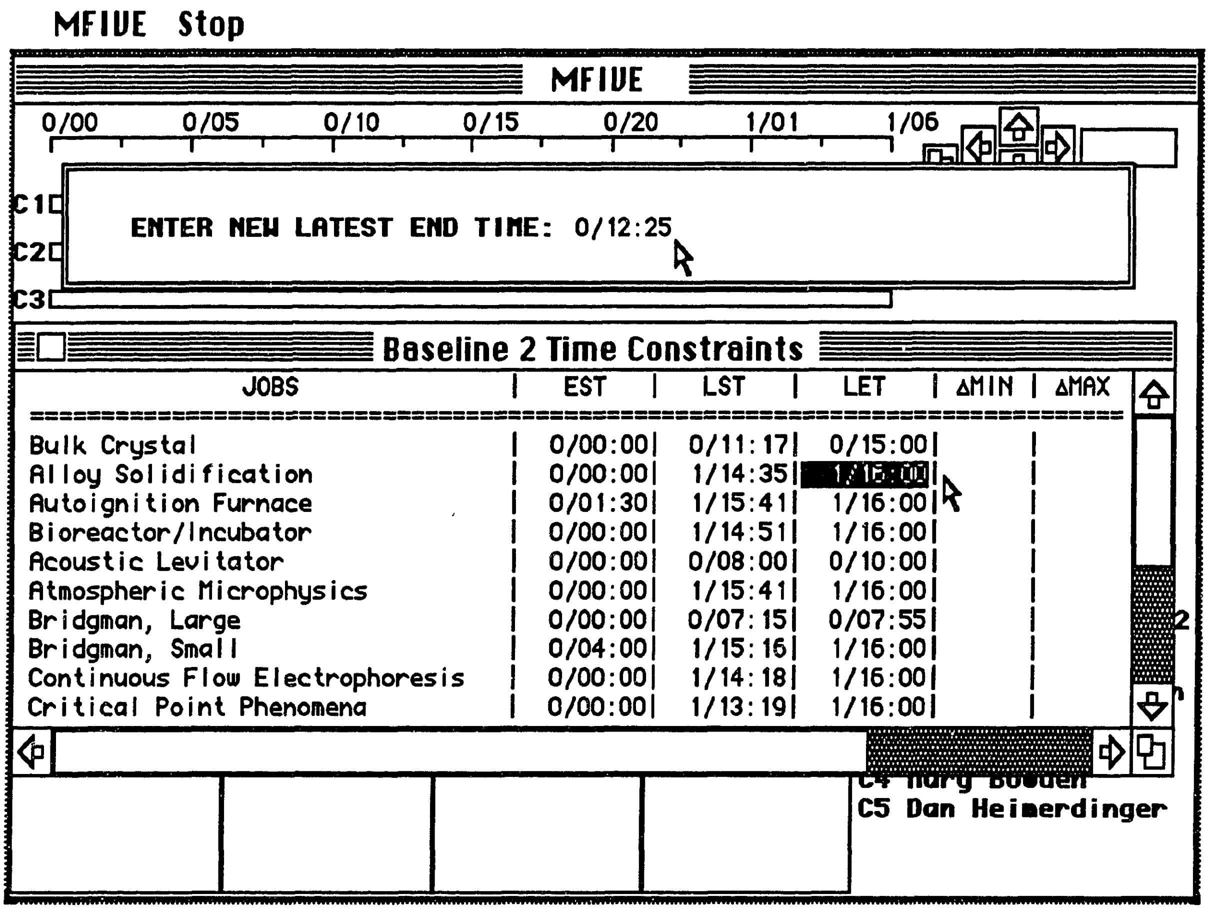


Figure 6.41: Modifying the Latest End Time of ALLOY SOLIDIFICATION

By clicking on ACOUSTIC LEVITATOR, the last four columns of the table get filled in, expressing time constraints which relate the other jobs to ACOUSTIC LEVITATOR (the first two columns of the table in Figure 6.42 have been scrolled off the screen). The fifth column, labelled ΔMIN, contains the minimum amount of time separating the start times of each of the jobs from the start time of ACOUSTIC LEVITATOR. For example, we see that the job

BULK CRYSTAL could start at most 0 days, 8 hours and 0 minutes before (as indicated by the negative sign) ACOUSTIC LEVITATOR. This would occur if BULK CRYSTAL were scheduled to start at the beginning of the flight plan and ACOUSTIC LEVITATOR were scheduled to start at its latest possible start time, 0/8:00.

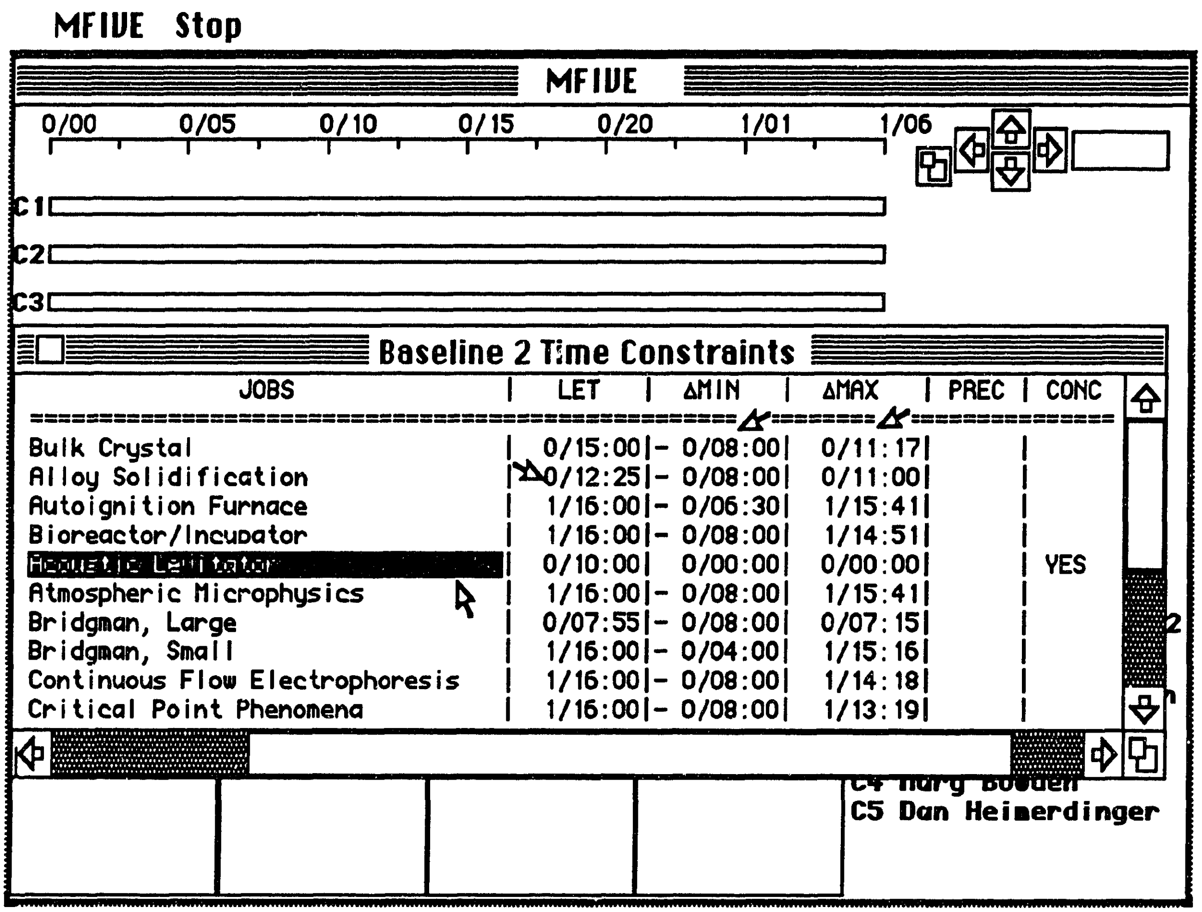


Figure 6.42: Time Constraints Relative to ACOUSTIC LEVITATOR

The sixth column, labelled ΔMAX, contains the maximum difference in start time between each of the jobs and ACOUSTIC LEVITATOR. For example, we see that the job BULK CRYSTAL can start at most 11 hours and 17 minutes after ACOUSTIC LEVITATOR. This would occur if ACOUSTIC LEVITATOR were scheduled to start at the beginning of the flight plan, and BULK CRYSTAL were scheduled to start at its latest possible start time, 0/11:17. Both ΔMIN and ΔMAX constraints can be modified by clicking on the appropriate entry.

The seventh column, labelled PREC, contains precedence constraints between each of the jobs and ACOUSTIC LEVITATOR. Figure 6.43 shows the process of adding a precedence constraint. The user has clicked on the space for a CRITICAL POINT PHENOMENA precedence constraint, and the system has responded with a window asking the user if CRITICAL POINT PHENOMENA should occur either before or after ACOUSTIC LEVITATOR. Figure 6.44 shows the result of selecting BEFORE. In addition to including the word BEFORE in the PREC column, it can be seen in Figure 6.44 that the latest end time of CRITICAL POINT PHENOMENA has been changed to 0/8:00, which is the latest start time of ACOUSTIC LEVITATOR. The latest start time for CRITICAL POINT PHENOMENA has also been similarly reduced.

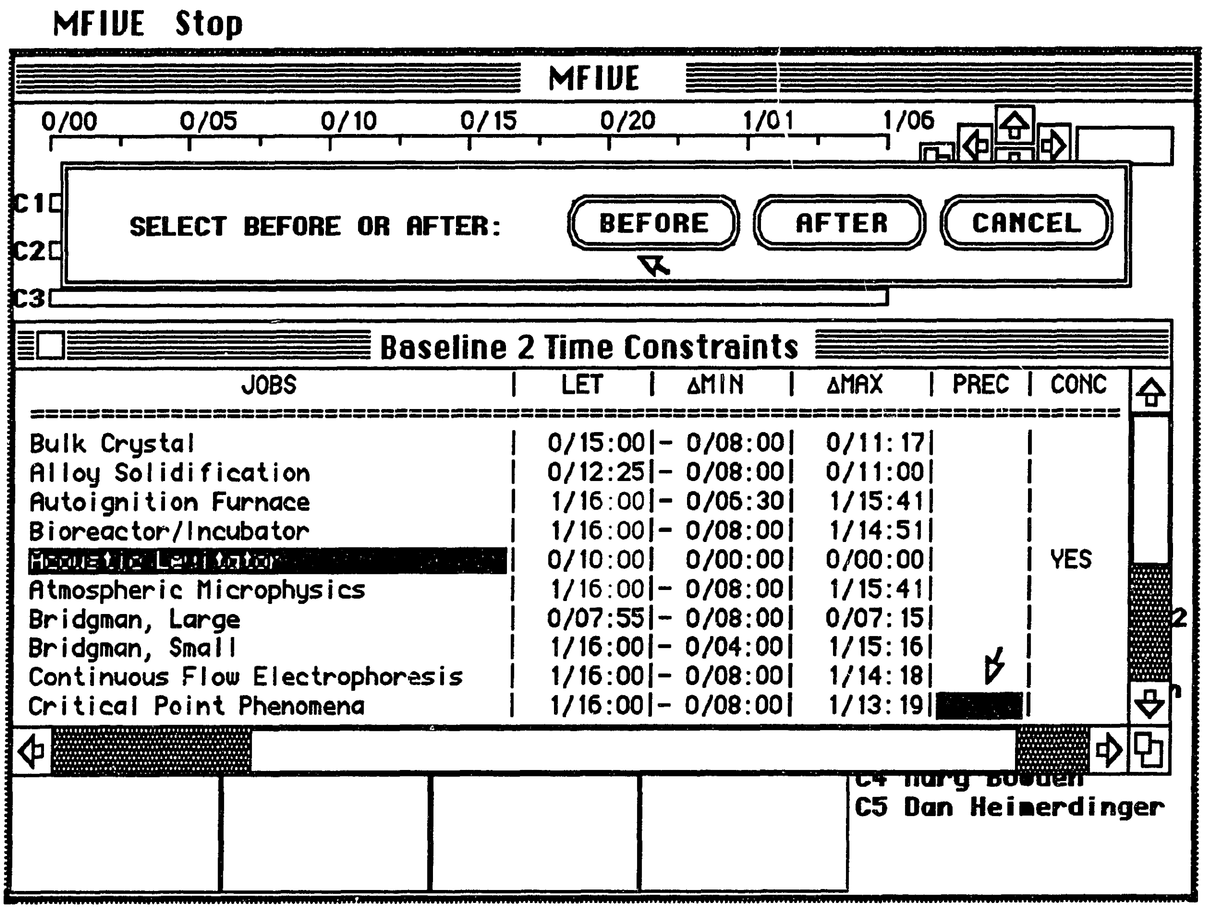


Figure 6.43: Requiring CRITICAL POINT PHENOMENA to Start Before ACOUSTIC LEVITATOR

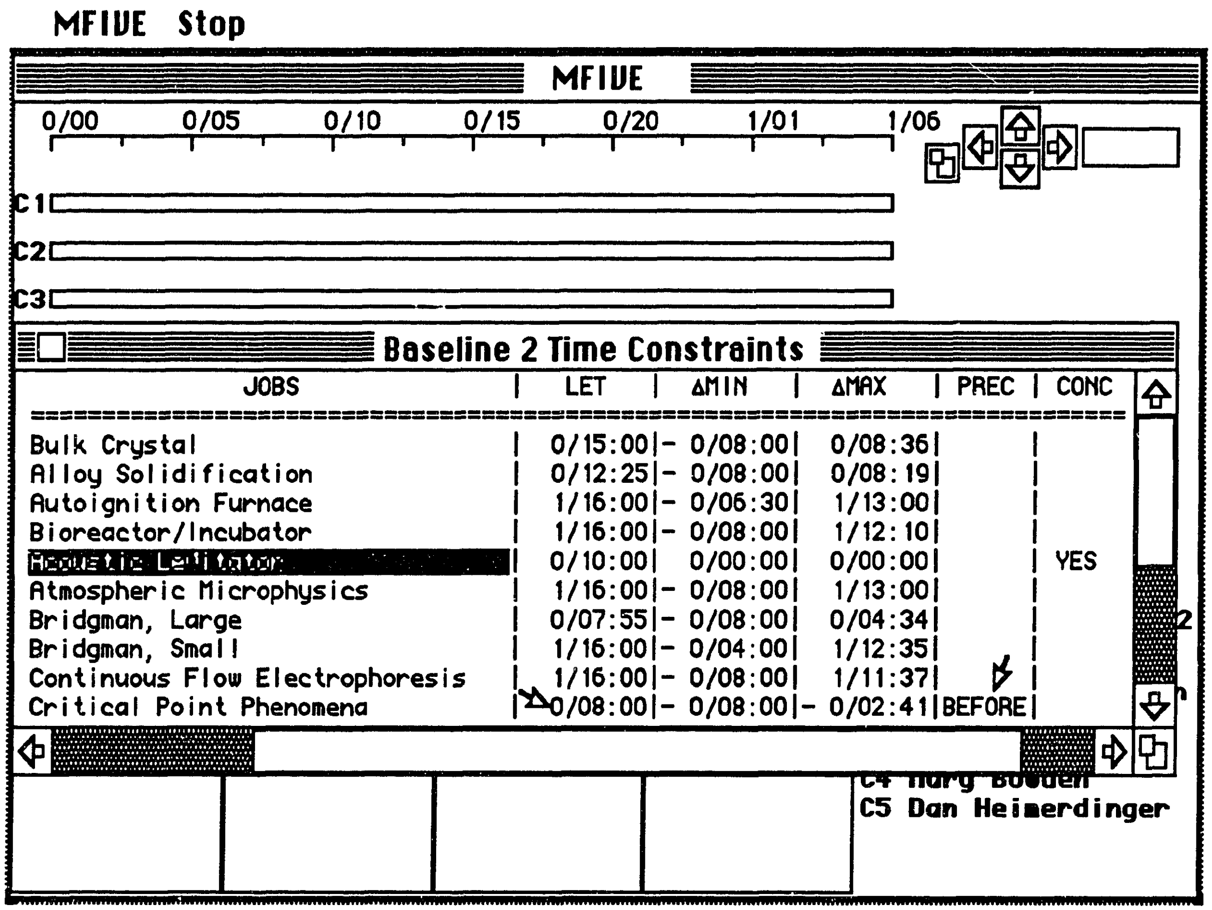


Figure 6.44: Results of Requiring CRITICAL POINT PHENOMENA to Start Before ACOUSTIC LEVITATOR

In general, the addition of any new constraint into the constraint table can cause propagation of changes effecting any of the other items in the table. The scheduler utilizes algorithms which make maximum possible inference during constraint propagation. These algorithms are fully discussed in Section 4.2.

The eighth column of the constraint table contains the concurrence constraints. Clicking on the space for a concurrence constraint allows the user to require the start times of any two jobs to be the same.

The system will not permit the user to enter mutually contradictory constraints. For example, after telling the system that CRITICAL POINT PHENOMENA is before ACOUSTIC LEVITATOR, one could not enter a concurrence constraint between those two jobs. It can also be seen that earliest start time and ΔMIN values must always increase, while latest start time, latest end time and ΔMAX values must always decrease, if constraint contradiction is to be avoided.

At present, there is no way for a user to delete a previously added constraint, other than to restart the scheduling of the flight plan. Providing this option is difficult because removing a constraint would in addition involve removing all of the inferences made from that constraint. In order to implement this capability, one would have to explicitly keep track of each time constraint as it is entered. An interface would also have to be provided to allow the user with a method of revising or deleting the constraints. Because any one constraint can effect the entire constraint table, there is no way to remove its effects other than by effectively restarting the constraint propagation (as is possible, using the RESTART SCHEDULE option from the AUTOPLAN menu, Figure 6.31). Beginning with a table not containing any time constraints, one would then have to re-apply each of the constraints.

6.2.9 Target Constraints

In actual operation, target constraints (i.e., time windows during which a task can or cannot be performed) can be specified by the satisfaction of a large number of varied functions, such as meeting a desired set of orbital parameters. While explicitly modeling each possible source of target constraints is an interesting problem which will have to be addressed in a fully operational system, it is not crucial to the scheduler's operation. From the point of view of the scheduler, it is only the final time windows which are relevant, and not the functions which produce them.

Selecting the TARGETS option from the CONSTRAINTS menu allows the user to specify windows during which performance of a job is not feasible. The user is first given a listing of jobs to select from (Figure 6.45). After selecting a job, a box appears (Figure 6.46) allowing the user to add, modify, or delete windows during which the job cannot be performed. (Those jobs with an asterisk have already been scheduled.) Figure 6.46 shows, for the job AUTOIGNITION FURNACE, its earliest start time, latest start time, latest end time, and two windows during which it has been specified as infeasible. Clicking on the number of a window will allow the user to delete it. Window start or end times can be revised by clicking on them. Scrolling the information to the bottom will allow the user to add additional windows. After closing this box (by clicking on "done") and selecting AUTOIGNITION FURNACE for scheduling (Figure 6.47), it can be seen that the infeasible windows have been blacked out, and the job therefore cannot be scheduled during these periods.

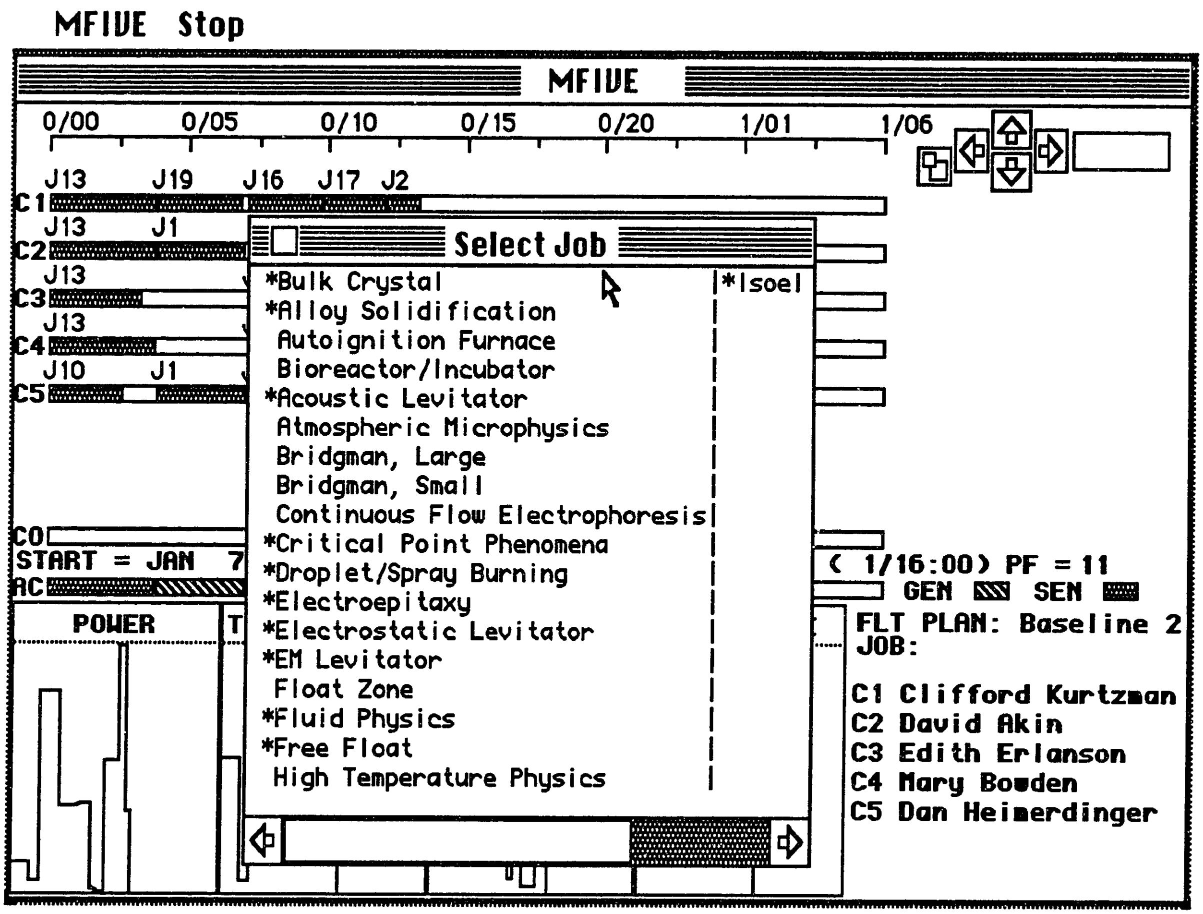


Figure 6.45: Selecting a Job for Setting Target Constraints

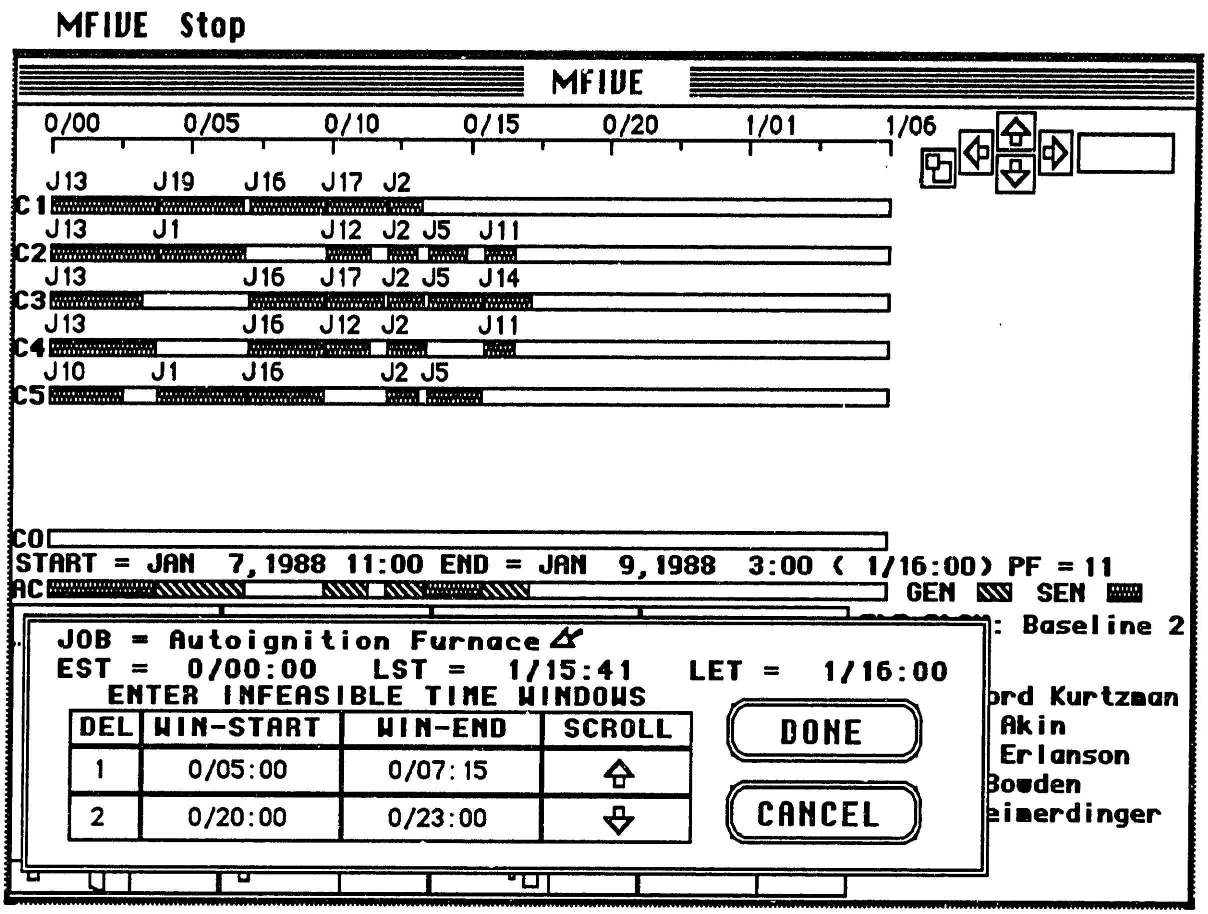


Figure 6.46: Target Constraints for AUTOIGNITION FURNACE

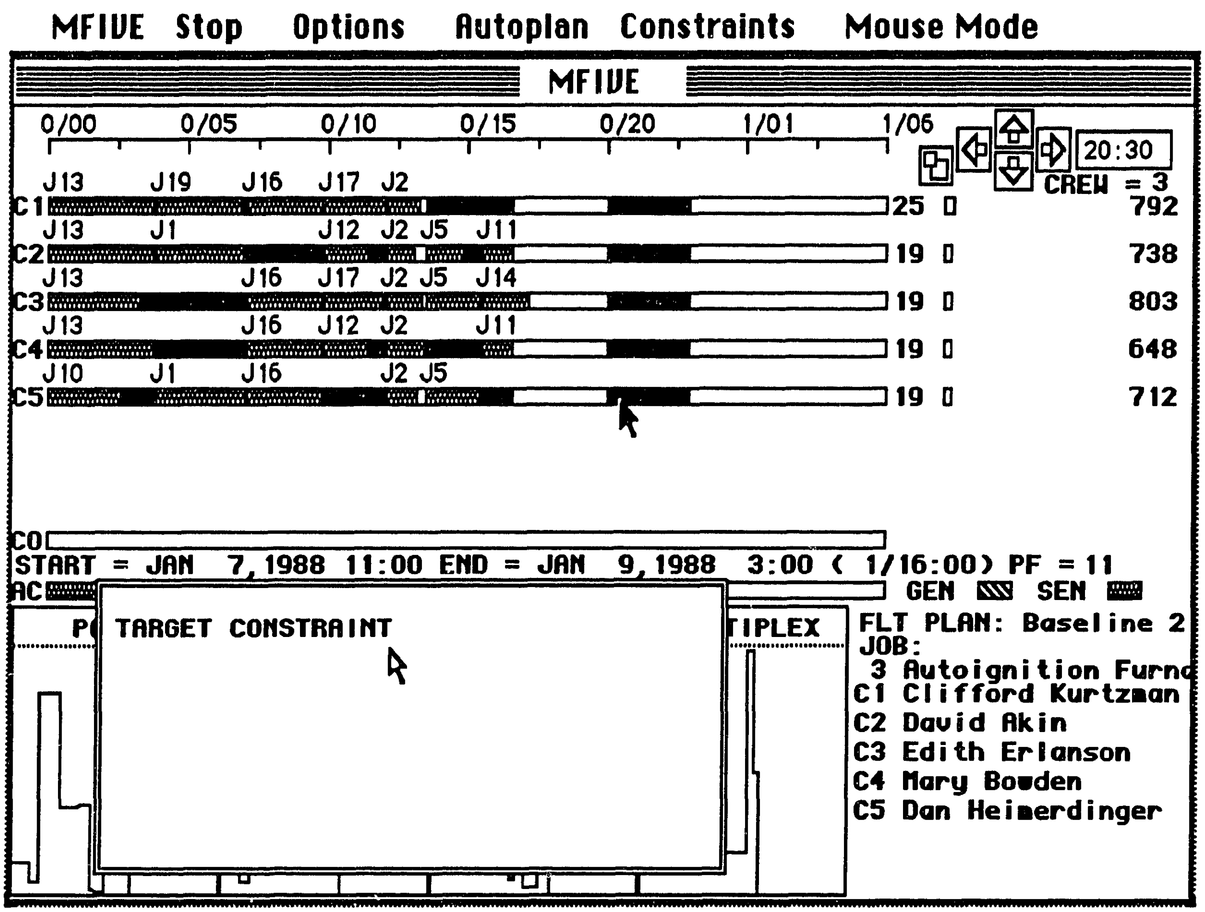


Figure 6.47: Display of Target Constraints for AUTOIGNITION FURNACE

6.2.10 Dynamic Rescheduling

Dynamic rescheduling allows the user to perform manual intervention in fixing jobs within a schedule, and to edit and revise previously determined schedules. The MOUSE MODE menu (Figure 6.48) contains several options for modifying a schedule. When a job is scheduled, its start time and crew assignment are designated as either "fixed" or "free." The start time is designated as fixed if it is explicitly specified by the person performing the scheduling. The start time is designated as free if it is chosen by MFIVE (for example, by using the AUTO CREW SELECT option). Similarly, the crew assignment of a job is said to be fixed if it was explicitly specified, and free if it was chosen by MFIVE. With the mouse mode set to the IDENTIFY JOB setting, clicking the mouse on the gray rectangle corresponding to a job displays a window (Figure 6.22) giving the job's name, and whether its start time and crew assignment are free or fixed.

MFIVE has facilities for allowing the user to modify the fixed or free status of a job. This is important because when a user has specified a partial schedule, the COMPLETE SCHEDULE option from the autoplan menu (see Section 6.2.4) can be used in order to let MFIVE complete the scheduling process. Using the COMPLETE SCHEDULE option will not alter start times or crewmembers assignments if they are fixed. Any start times or crewmember assignments which are free, however, are subject to modification during the search for a better schedule.

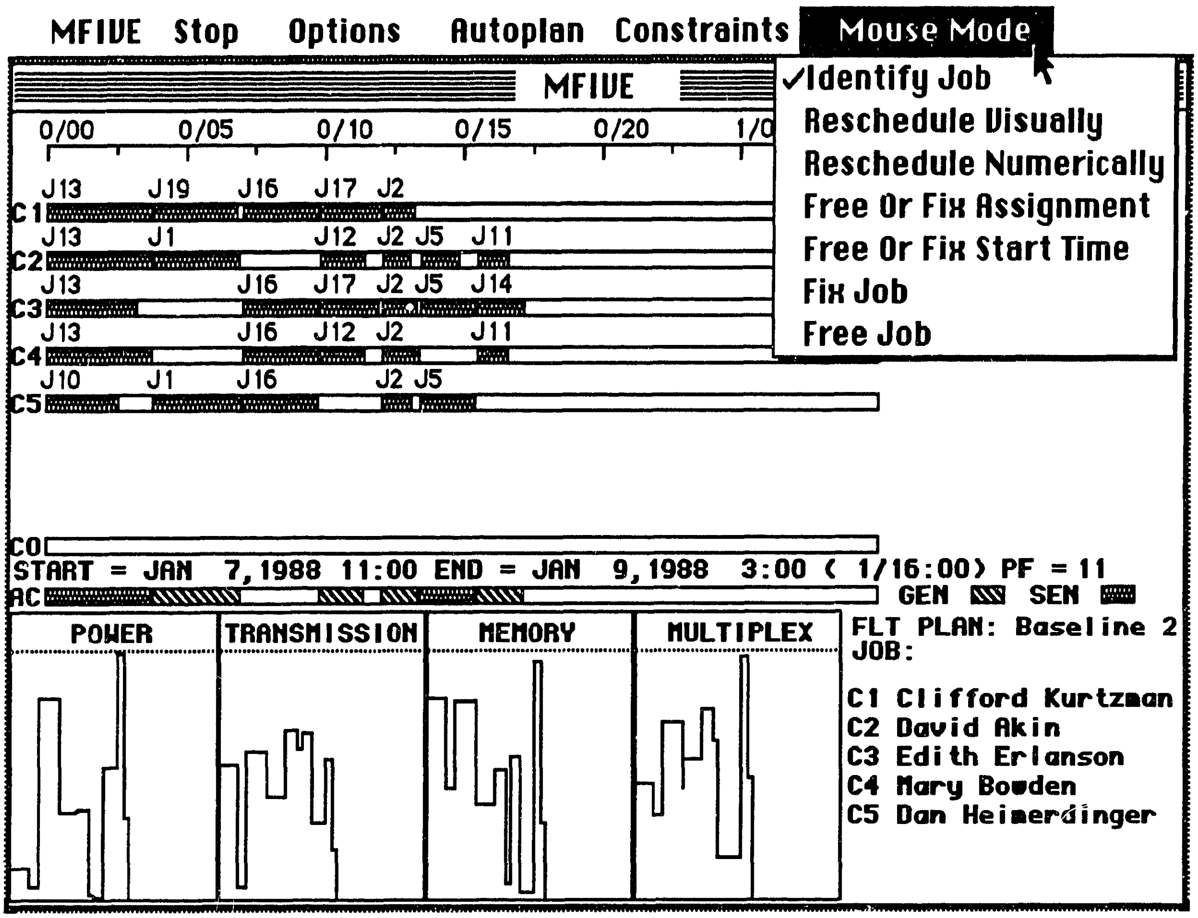


Figure 6.48: The MOUSE MODE Menu

The MOUSE MODE menu includes options for using the mouse to set crewmember assignment and the start time to either the fixed or free state. In addition, by pulling down the RESCHEDULE NUMERICALLY option and then clicking on a job, the user is presented with a form (Figure 6.49) which allows the job's assignment and start time to be explicitly modified. Using this form, the user can change the start time and set whether it is fixed or free. It is also possible to change the crewmembers who are assigned to the job, or to unschedule the job completely.

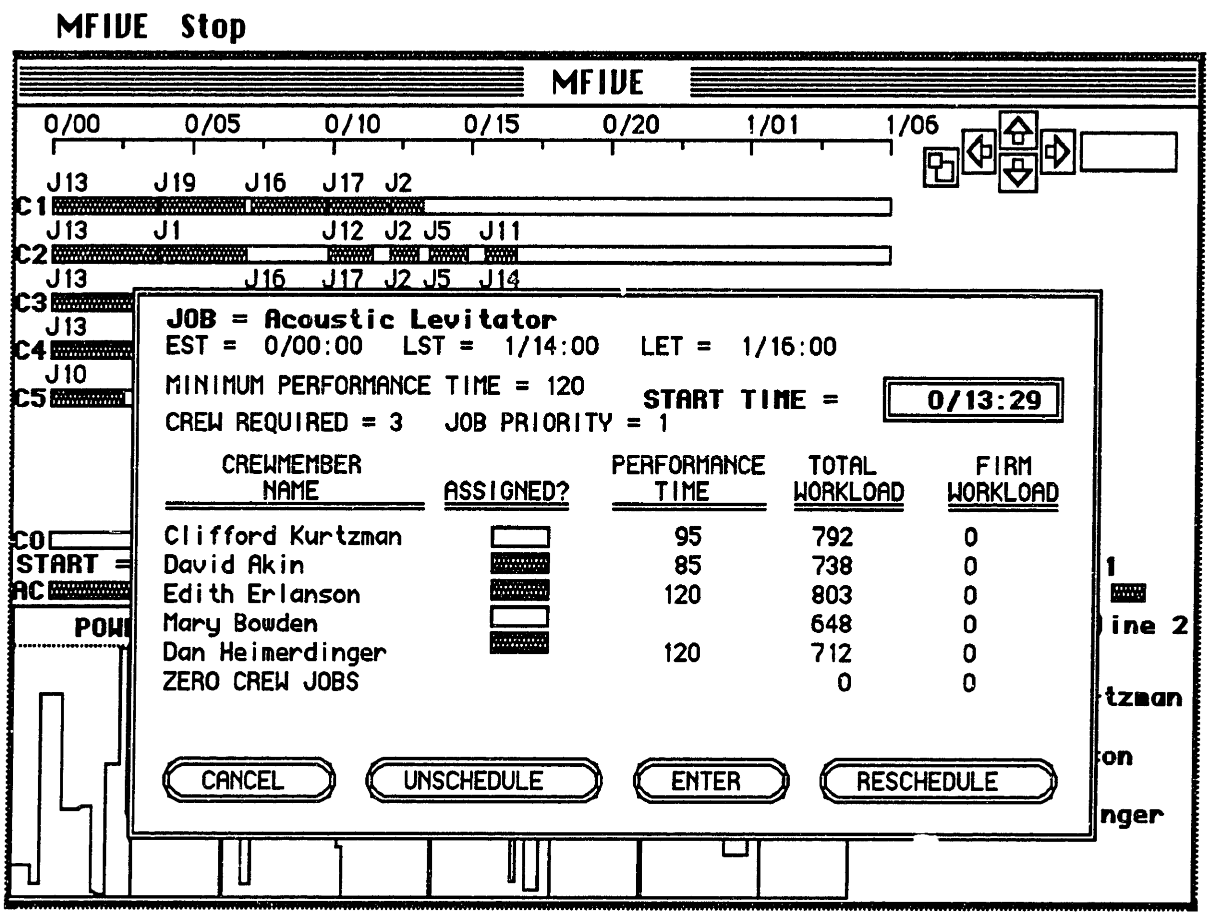


Figure 6.49: Rescheduling Form for the Job ACOUSTIC LEVITATOR

6.2.11 Suggestions For Future Development

Operational experience with the scheduler has indicated many areas in which improvements could be implemented. One major problem with the present scheduler implementation is the limitation caused by keeping the primary database separate from the scheduler itself. To an extent, it is rather arbitrary that requirements relating to resource constraints are specified in the data base management system, which those relating to time constraints are specified in the scheduler. The primary distinction between these types of constraints is that resource characteristics relate to individual jobs, whereas time constraints relate how jobs interact within the context of a specific flight plan. As a consequence of this separation between the data base management system and the scheduler, there is no way in which job characteristics (such as performance time or resource requirements) can be changed once the user has entered the scheduler. It is also not possible to add new jobs to a flight plan once it has been sent to the scheduler. The only way to accomplish any of these functions is to go back to the data base managment system, modify the appropriate job information form, and then branch back to the scheduler. Scheduling must then be restarted, and all time and target constraints must be reentered. As was pointed out in Section 6.2.8, there is at present no method for reapplying (or editing) a previously entered set of time constraints, other than respecifying them one by one. An interface which could perform this function would be difficult to build, but would prove very helpful to the scheduling process.

The inability to change job data inside the scheduler is a drawback because experience shows that the process of scheduling is inherently tied to the process of manifesting (defining job attributes and selecting a group of mutually compatible jobs to be scheduled together). In the real world, typical experience shows that defining a schedule is an iterative process where one choses a set of jobs which one hopes will schedule together. One then attempts scheduling, but may find that some jobs are completely incompatible with others (MFIVE may be very efficient at finding good schedules when they exist, but it is only as good as the data it is fed). This is dealt with by either removing some of the incompatible jobs, by changing their characteristics, or by relaxing some constraints so that they might be made schedulable. For example, a mission planner might go to a scientist planning an experiment and ask (or demand) that it be shortened by ten minutes to allow for the accomplishing of other mission priorities. Another possible scenario would be to decide to delay some experiment to a later mission.

Alternatively, preliminary scheduling runs might indicate that large gaps of free time exist in crewmember schedules, and therefore it would be advisable to add additional experiments.

Several complications would be imposed in trying to provide a capability to interactively schedule and alter the basic job data. During scheduler operation, many inferences are made from the job data (e.g., in the course of constraint propagation), which would have to be revised if the job data were changed. As explained in Section 6.2.8, a large amount of computation may be necessary in order to regenerate these inferences. A change to the job data could also cause infeasibilities to occur. For example, increasing the length of a job could cause it to become too long to fit within its feasible time window.

As was also mentioned in Section 6.2.8, it would be helpful if there were a method to edit or remove time constraints which have been added to a schedule. Providing a good interface for doing this would involve a large programming effort.

Another area of scheduler operation which could significantly benefit from improvement is the use and modelling of resource constraints. The present implementation of MFIVE only accommodates four linear, non-time varying resources, and the audio/vibration constraint. As described in Section 3.3.1, the real space station environment contains numerous other types of constraints, such as nonlinear constraints, jobs with time varying resource usages, time varying resource limits, resource limits which are a function of when jobs are scheduled, and non-renewable resources. None of these constraint types should present any fundamental difficulties to the scheduler, but providing an interface to allow a user to create and enter these types of constraints is a significant programming problem. More complicated resource constraints will also slow scheduler operation.

MFIVE operation showed the need for several types of constraints which were initially unanticipated. A concurrence constraint can be used to require the simultaneous execution of two jobs, but there is no type of constraint which will expressly prohibit simultaneous execution. One can specify that one job must precede another job or that it must follow another job, but there is no way to require that it either precede or follow. Another type of constraint which is needed would be to require that the crewmembers which are assigned to accomplish one job must also be assigned to another job. For example, one should be able to specify that those crewmembers which perform the EVA prebreathe also perform the EVA. An additional class of constraints which would be useful are "optional" time constraints. An example of this type of constraint would be to say "if there is going to be an exercise period, then it must be at least 90 minutes before dinner." The current version of MFIVE only allows "hard" constraints of the type "the exercise period must be at least 90 minutes before dinner."

There are several features that will be necessary for real time scheduler operation, but which are not yet implemented. For example, after generating and attempting to implement a schedule, it will be necessary to advise the system of those jobs which were successfully completed and those which were not, and might therefore require rescheduling later. If some job takes longer than expected, it might be necessary to start other jobs later than anticipated. There should be a capability to advise the system of this and easily see what impact this will have on the schedule (e.g. whether it will cause other jobs to become infeasible, or whether it might be advisable to alter the scheduling of other jobs).

In general, one should be able to ask the scheduler what the effect of a change will be. Continuing the above example, suppose it appears that a job will take longer than expected and therefore cause some other job to be started later or even not performed at all. One should be able to ask MFIVE to compare the relative qualities of schedules in which the job is allowed to run overtime (thus impacting other jobs) and schedules where the job is terminated uncompleted, returning to the baseline schedule. Another type of choice which the scheduler might be asked to make is to compute what penalty in schedule quality would be incurred by requiring a job to start at a specific time, as opposed to allowing MFIVE to autonomously decide what starting time is best. A useful capability would be to have the MFIVE insert a job into a schedule, making the necessary adjustments on other jobs but keeping these adjustments as small as possible. It would also be helpful to be able to move a group of jobs *en masse* to another time in the schedule. Providing sophisticated options such as these will be quite computationally intensive, and will probably have to await scheduler implementations in faster computer languages and on faster computers.

6.3 The Tool Searcher

Activating the search button on the Tool Information Form (Figure 6.14) will initiate a search for a missing tool. Figures 6.50 and 6.51 shows the results of performing a search for the Cable. The default locations are shown, which indicate where the tool *should* be found. After the default locations is a ranked listing of likely places to look for the tool. Compartments are ranked based on numerous factors such as compartment size (in relation to the tool's size), where the tool was last used, the pathway from where the tool was last used to where the tool belongs, and where the tool has historically been found. Also given is the name of the crewmember who last used the tool. In his Masters thesis, Kranzler discusses in greater depth the working and implementation of the tool searcher [Kranzler, 1986].

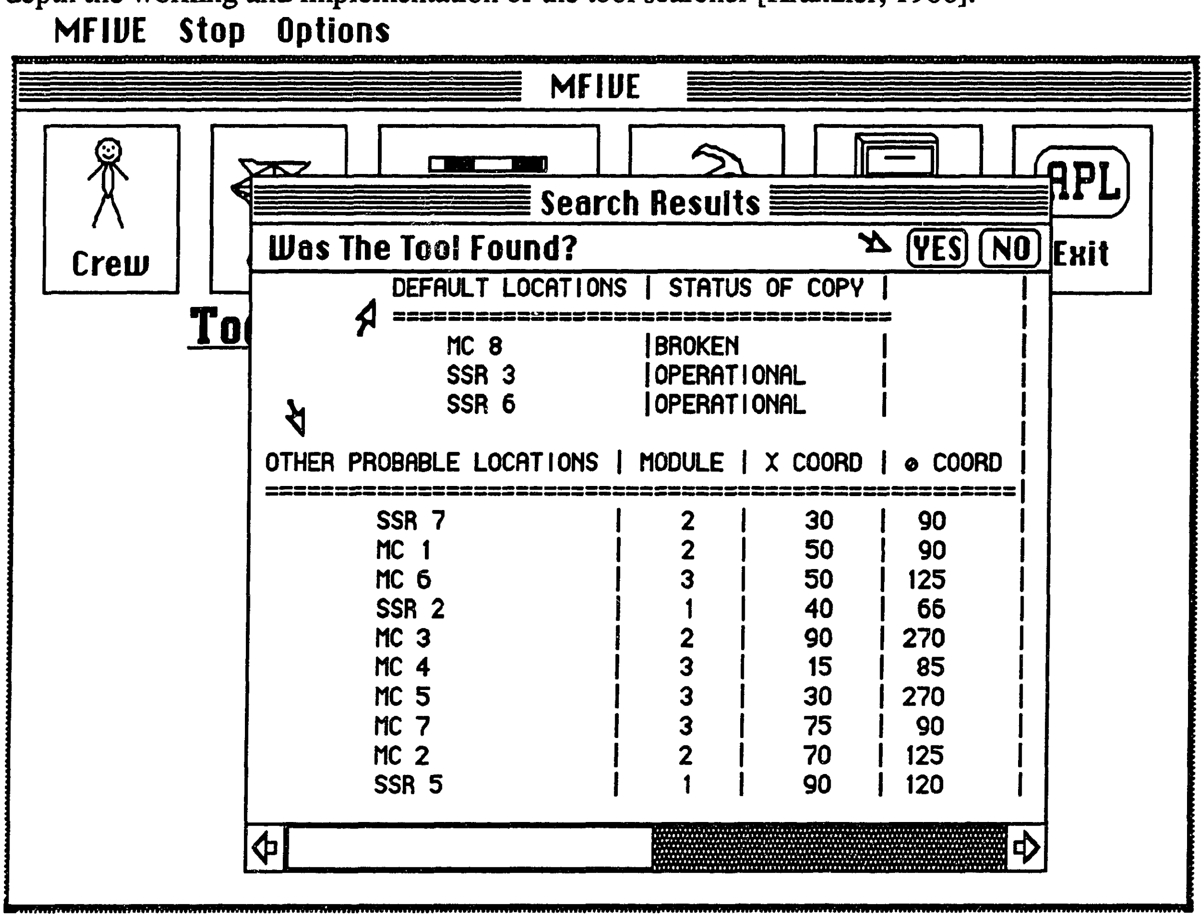


Figure 6.50: Results of Performing a Search for the CABLE

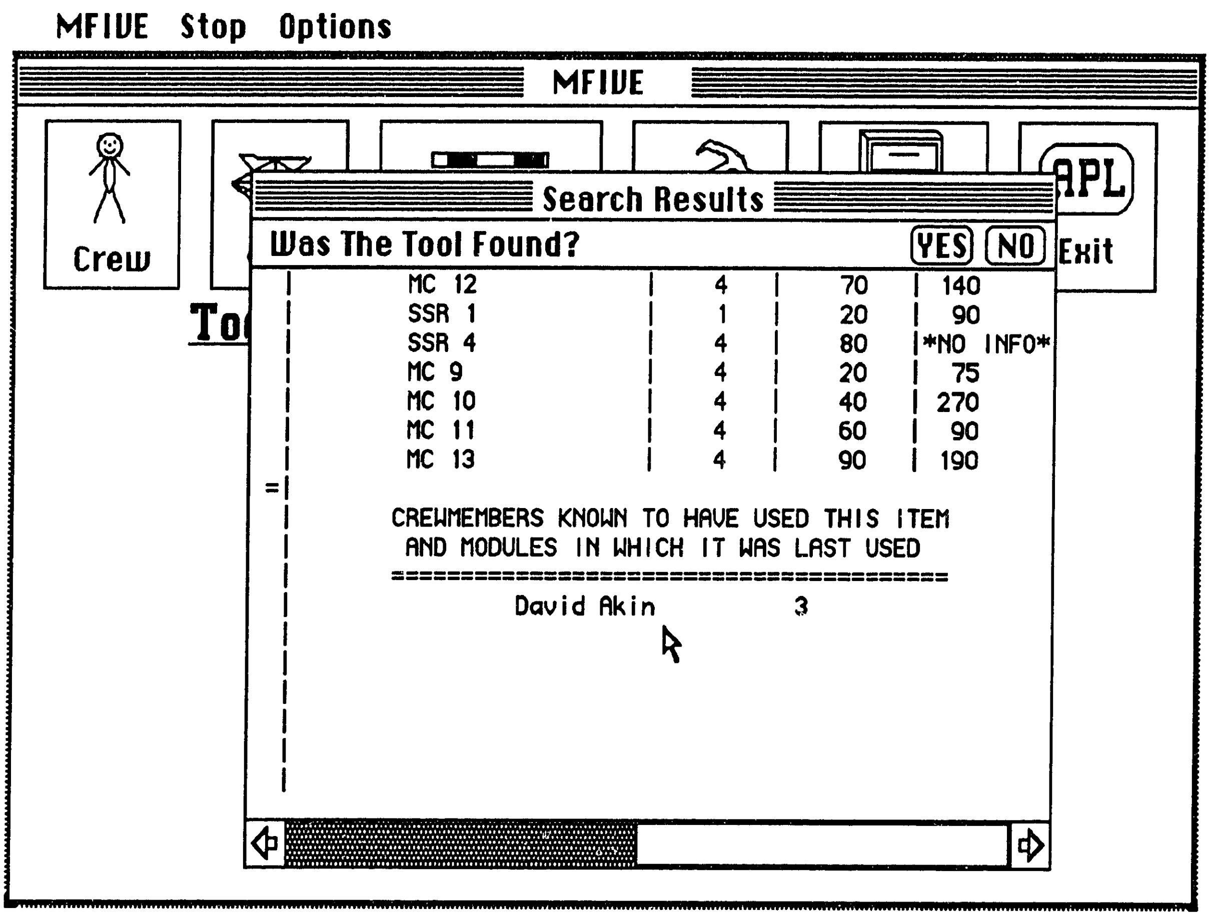


Figure 6.51: Performing a Search for the CABLE

The form presented in Figure 6.50 ask the user whether or not the tool was successfully found. If it was, then the user is asked to identify the compartment in which it was found (Figure 6.52). The searcher then notes this information and increases the ranking of that compartment in subsequent searches for the tool.

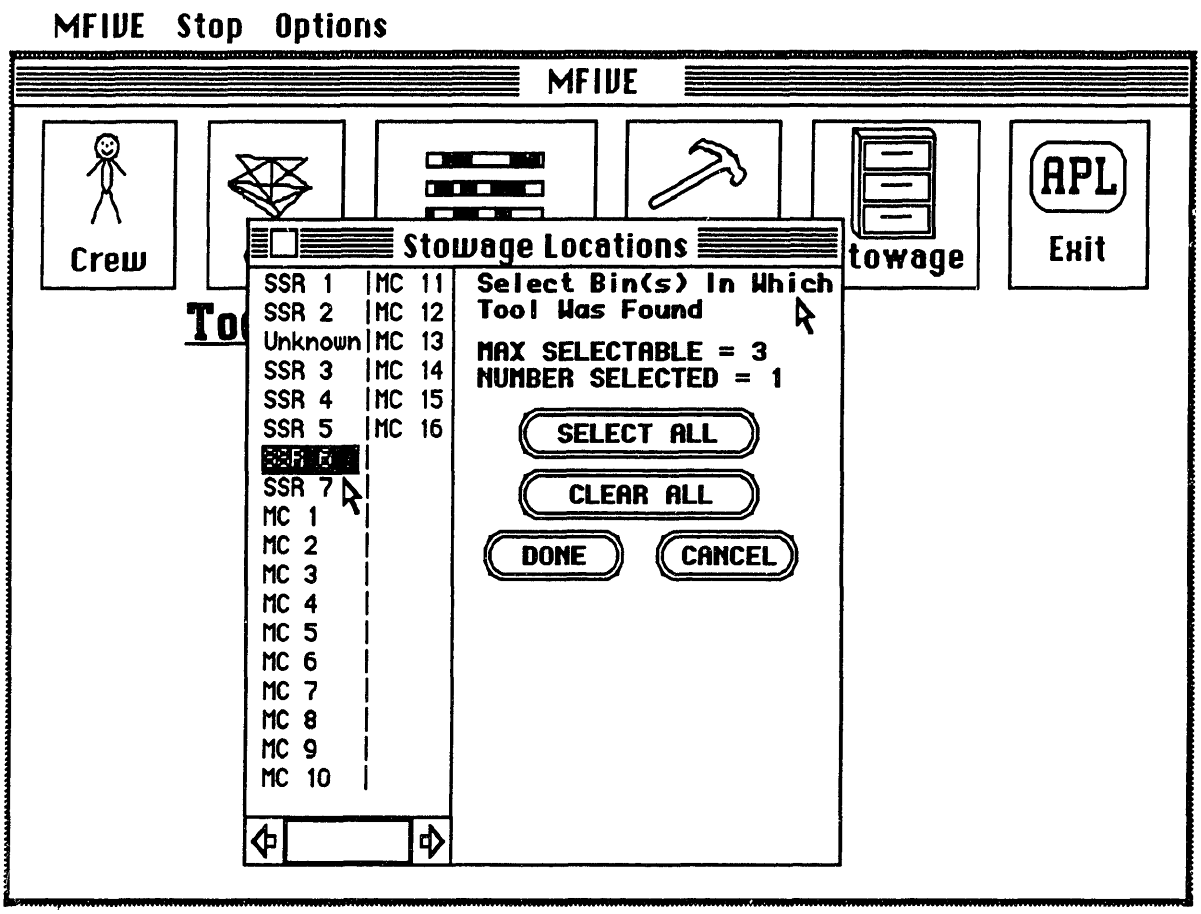


Figure 6.52: Indicating that the CABLE was Found in SSR 6

It should be noted that the present implementation of the tool searcher contains many features which are simulated. It is envisioned that eventually scheduling infomation produced by the scheduler would be accessible to the searcher, so that it could get factual infomation about where and by whom jobs were performed. In order to implement this, it will be necessary to add an interface to the Job Information Form specifying where each job is performed. It will also be necessary to add a feature to the scheduler so that a user can give feedback as to when and if a proposed schedule was actually carried out.

The tool searcher also assumes that the space station has a configuration which is essentially hard-wired into MFIVE. There is at present no facility for a user to add or delete modules, or to specify the connectivity of the modules.

Section 7: Conclusions and Recommendations

Crew activity planning is a complex problem whose solution will be critical to permit efficient space station operation in the 1990's. The methodologies developed in this thesis will hopefully contribute towards that goal. A technique has been described for propagating complex sets of time constraints to make inferences about feasible windows for the scheduling of jobs which compete for limited resources and crewmember availability. The model developed can accommodate time constraints which relate to fixed points in time, such as earliest start time constraints, latest start time constraints, and latest end time constraints, as well as constraints which relate jobs to other jobs in a schedule, such as precedence constraints, concurrence constraints, and constraints specifing maximum and minimum time intervals between the start times of different jobs. The model can also accommodate uncertainties in the durations of jobs. A simple example would be if there were a constraint requiring Job B to start at most 5 hours after Job A, a constraint requiring Job C to start at most 3 hours after Job B, and a final constraint specifying that the latest start time of Job A was at 2:00 a.m. One could then make the inference that Job C could start no later than 10:00 a.m.

In practice, when there are a large number of these different constraints present, cross-relating a large number of jobs, the problem of making inferences of this type is not so trivial. The standard critical path method deals with jobs of known duration in scenarios usually involving only precedence constraints. This thesis (Section 4.2) shows how the information contained in the wide variety of constraints described above can be used to form a shortest path problem, which can be solved to make maximum possible inference to narrow down the feasible time window for each job. This information makes substantial headway in reducing the complexity of the entire scheduling problem being solved. It is particularly valuable in scenarios where a person is interactively scheduling jobs on a computer; it is possible to visually show the person performing the scheduling the time window in which the job can be scheduled, while still permitting sufficient room for all the other jobs which are related to it by time constraints.

While it is desirable to give human schedulers as much freedom and flexibility as possible while performing the scheduling process, the problem of efficiently scheduling hundreds of complex tasks is too difficult and complicated to be done in a reasonable amount of time solely by human operators. Even with the best computational tools, which can propagate time constraints (as described above) and show a human scheduler feasible time windows and resource conflicts, the problem is still too difficult. An ideal means of operation would be for a human operator to specify initial preferences, and then let a computer autonomously plan out the rest of a schedule. The human operator could then interactively edit the computer generated schedule and make any desired changes. In this way, full use would be made of the computer's ability to process large quantities of numerical data in order to find an "optimal" solution, making best use of crewmember time and space station resources.

In order to facilitate autonomous computer solution of scheduling problems, a new technique was developed in this thesis, known as "the iterative algorithm," (Section 5.2.4) for finding good solutions to the crew activity planning problem. This algorithm is a dispatching algorithm which selects jobs for scheduling one at a time, in order of decreasing priority, after which they are added to a partial schedule. After each attempt at scheduling the jobs, the iterative algorithm attempts to identify the jobs which are causing difficulties in producing a better schedule. The priorities of these jobs are then increased so that they are scheduled earlier on the next iteration of the algorithm.

A number of heuristic rules were also used to determine the relative priorities of the jobs, and the relative performances of each of these rules were then evaluated on several test problems. For example, a Longest Job First Heuristic would assign a priority to each job proportional to the job's duration. The total priority for any given job could then be calculated by multiplying its heuristic priority by its priority given through successive iterations of the iterative algorithm. A new heuristic technique, the Maximum Compatibility Method, was developed (Section 5.1.3) for this thesis. This technique is different than heuristics such as the Longest Job First Heuristic, which maintain data on the relative importance of each job in the dispatched ordering. In contrast, the Maximum Compatibility Method maintains data on the relative importance for each pair of jobs to be dispatched one after the other. Schedules are then generated by dispatching jobs in an order which will produce a large degree of compatibility between jobs which are scheduled sequentially, thus allowing a large amount of overlap between jobs in the schedule which is produced.

For each of the heuristics tested, two versions were tried. These two versions are named the "deterministic version" of the heuristic and the "randomized version" of the heuristic. For example, in the deterministic version of the Longest Job First Heuristic, jobs would be dispatched for scheduling in order of decreasing job length. In the randomized version of the Longest Job First Heuristic, at each stage in the scheduling process each remaining unscheduled job would have a probability of being dispatched which would be proportional to its length. Longer jobs would have a greater probability of being scheduled first, but would not be guaranteed to be scheduled first.

As described in Section 5.3, excellent results were obtained from using the iterative algorithm. For a typical problem, direct (non-iterated) application of the heuristics produced solutions on the order of 20% worse than the optimum solution. After performing 100 iterations using the iterative technique, the solution could be improved to (on average) only about 7% worse than optimum using a standard heuristic, and about 4% worse than optimum using the Maximum Compatibility Method. This is a significant improvement.

It was also found that while the deterministic versions of an heuristic usually produced better solutions on early iterations of the iterative algorithm, better final solutions (after 100 iterations) were usually produced by the randomized versions of the heuristics. The randomized versions were better able move away from local minima in the search process and thereby find better solutions. It was thus recommended that a hybrid technique be used whereby an heuristic would operate deterministically during early iterations, but operate in the randomized fashion during later iterations.

One additional attribute of the iterative algorithm is its ability to operate on problems in which the direct (non-iterated) heuristic would fail completely. For problems in which there are many complicated time and resource constraints, dispatching the jobs in the ordering indicated by a heuristic such as the Longest Job First Heuristic might lead to a situation where some of the jobs cannot be feasibly accommodated into the schedule. The iterative algorithm will search for new orderings which accommodate more and more of the jobs into the schedule.

In order to demonstrate the propagation of time constraints and the iterative algorithm, an interactive computer scheduling tool, known as the MFIVE Crew Activity Planner, was developed (Section 6). MFIVE provides a convenient and user friendly interface for building, solving, and displaying scheduling problems, as well as for investigating the features which will be necessary to eventually provide a real-time scheduler for use on a space station.

Research into the iterative algorithm would greatly benefit from the testing of larger, more complex problems than was performed for this thesis. The current version of MFIVE cannot accommodate problems with more than about 100 jobs. This limit is placed both by the amount of memory available and the speed of the Macintosh computer. For a typical problem with 20 jobs, it takes MFIVE about 5 seconds to add each job to a schedule. This includes choosing the job, applying the pertinent time and target constraints, finding the best place to insert it into the schedule, inserting the job into the schedule, propagating the effects of the job on other jobs, and updating the screen. This translates to about 100 seconds for a single iteration of the iterative algorithm. This is much slower than is desired. A speed gain of perhaps a hundredfold could be achieved by using a faster, compiled, computer language. Additional speed enhancement will also be possible by using faster computers. No extreme attempt has been made to streamline the computer programs to find the most efficient data structures and means of implementation, and there is no doubt that further gains in speed are also achievable in this area.

A next generation version of MFIVE (MSIX?) should be able to incorporate many types of capabilities not present in the current version. As discussed in Section 6.2.11, this would include the ability to model more complex resource constraints and perform more sophisticated interactive editing during schedule construction.

In summarizing, it is believed that the techniques and results generated in this thesis will prove useful in the ultimate development of a system for performing space station crew activity planning. With the ability to generate efficient schedules either autonomously or interactively, such a system will enhance space productivity and enhance crewmember morale, to thereby allow crewmembers to make maximum use of the space environment for performing mission-oriented activities.

Bibliography

Altman, S.M.; Beltrami, E.J.; Rappaport, S.S.; Schoepfle, G.K.: Nonlinear Programming Model of Crew Assignments for Household Refuse Collection. IEEE Transactions on Systems, Man, and Cybernetics, July, 1971, pp. 289-291.

Balbaky, E.M.L.: Strike in Space. Case #1-431-008, Harvard Business School, Boston Massachusetts, 1980.

Bensana, E.; Correge, M.; Bel, G.; Dubois, D.: An Expert-System Approach to Industrial Job-Shop Scheduling. Proceedings of the 1986 IEEE International Conference on Robotics & Automation, April 7-10, 1986, pp. 1645-1650.

Blagov, V.D.: Deputy Flight Director Blagov on 211-Day Flight of Berezovoy and Lebedev. USSR Report: Space, June 14, 1984. JPRS-USP-84-003.

Blazewicz, J.; Lenstra, J.K.; Rinnooy Kan, A.H.G.: Scheduling Subject to Resource Constraints: Classification and Complexity. Stichting Mathematisch Centrum, Amsterdam, Department of Operations Research, August, 1980.

Brand, J.D.; Meyer, W.L.; Schaffer, L.R.: The Resource Scheduling Problem in Construction. Civil Engineering Studies, Report No. 5, Dept. of Civil Engineering, University of Illinois, Urbana, 1964.

Conway, R.W.; Maxwell, W.L.; Miller, L.W.: Theory of Scheduling. Addison-Wesley, Reading, Massachusetts, 1967.

Cooper, H.S.F., Jr.: A House in Space. Holt, Rinehart and Winston, New York, 1976.

Cooper, D: Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation. Management Science, Vol. 22, No. 11, July, 1976, pp. 1186-1194.

Davies, E.M.: New Criterion for Resource Scheduling. Transportation Engineering Journal, Vol. 99, No. TE4, November, 1973, pp. 741-755.

Davis, E.W.: An Exact Algorithm For The Multiple Constrained-Resource Project Scheduling Problem. Unpublished Ph.D. Thesis, Yale University, 1969.

Davis, E.W. (1973a): Project Scheduling under Resource Constraints -- Historical Review and Categorization of Procedures. AIEE Transactions, Vol. 5, No. 4, December, 1973, pp. 297-313.

Davis, E.W. (1973b): Project Summary Measures and Constrained Resource Scheduling. Working Paper HBS 73-74, Graduate School of Business Administration, Harvard University, 1973.

Davis, E.W.; Heidorn, G.E.: An Algorithm for Optimal Scheduling Under Multiple Resource Constraints. Management Science, Vol. 17, No. 12, August, 1971, pp. B803-816.

Davis, E.W.; Patterson, J.H.: A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. Management Science, Vol. 21, No. 8, April, 1975, pp. 944-955.

Dempster, M.A.; Lenstra, J.K.; Rinnooy Kan, A.H.G.; eds.: Deterministic and Stochastic Scheduling. NATO Advanced Study Institutes Series. Series C: Mathematical and Physical Sciences, Vol. 84. D. Reidel, 1982.

Deuermeyer, B.L.; Shannon, R.E.; Underbrink, A.J., Jr.: Creation of the Selection List for the Experiment Scheduling Program (ESP). Industrial Engineering Division, Texas Engineering Experiment Station, Texas A&M University, College Station, Texas. Final Report of NASA Constract No. NAS8-35972, NASA-CR-178861, May 15, 1986.

Engelman, C.; Millen, J.K.; Scarl, E.A.: KNOBS: An Integrated Planning Architecture. AIAA Computers in Aerospace Conference, Hartford, CT, October, 1983, pp. 450-462.

Erschler, J.; Esquirol, P.: Decision-Aid in Job Shop Scheduling: A Knowledge Based Approach. Proceedings of the 1986 IEEE International Conference on Robotics & Automation, April 7-10, 1986, pp. 1651-1656.

Fisher, M.L.; Jaikumar, R.: An Algorithm for the Space-Shuttle Scheduling Problem. Operations Research, Vol. 26, No. 1, January-February, 1978, pp. 166-182.

Fox, M.S.; Allen, B.P.; Smith, S.F.; Strohm, G.A.: ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling. Intelligent Systems Laboratory, Carnegie-Mellon University, Pittsburgh, PA, June 21, 1983.

Freedman, D.H.: PC Software Helps Projects Run Smoothly. High Technology, May, 1985.

Gevarter, W.B.: An Overview of Expert Systems, NASA Headquarters, May, 1982. NBSIR 82-2505.

Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G.: Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Ann. Discrete Math., Vol. 5, 1979, pp. 287-326.

Grenander, S.: Toward the Fully Capable AI Space Mission Planner. Aerospace America, Vol. 23, No. 8, August, 1985, pp. 44-46.

Grone, R.D.; Mathis, F.H.: A Ranking Algorithm for Spacelab Crew and Experiment Scheduling. 1980 NASA/ASEE Summer Faculty Research Fellowship Program. N81-11986, October, 1980.

Hankins, G.B.; Jordan, J.W.; Katz, J.L.; Mulvehill, A.M.: Expert Mission Planning and Replanning Scheduling System. The MITRE Corporation, Bedford, Massachusetts, September, 1985.

Hayes-Roth, F.; Waterman, D.A.; Lenat, D.B.; eds.: Building Expert Systems. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1983.

Hitz, F.R.: Payload Crew Activity Planning Integration. Task 2: Inflight Operations and Training for Payloads. Contract NAS9-14676. Martin Marietta Corp., Denver, Colorado, N77-18739, December 23, 1976.

Hopfield, J.J.; Tank, D.W.: "Neural" Computation of Decisions in Optimization Problems. Biological Cybernetics, Vol. 52, 1985, pp. 141-152.

Hopfield, J.J.; Tank, D.W.: Collective Computation in Neuronlike Circuits. Scientific American. Vol. 257, No. 6, December, 1987, pp. 104-114.

Ivakhnov, A.: Changes in 'Soyuz T-10' Crew Work Schedule. USSR Report: Space, June 14, 1984, p. 18. JPRS-USP-84-003. Translated from Izvestiya, Feb. 17, 1984.

Jacubowicz, O.: Automatic Mission Planning & Scheduling Expert System (AMPASES). Status Report, GE/TRW, October 30, 1985

Jardine, T.J.; Shebs, S.T.: Knowledge Representation in Automated Mission Planning. AIAA Computers in Aerospace Conference, Hartford, CT, October, 1983, pp. 9-14.

Johnson, R.D.; et al: Autonomy and the Human Element in Space: Final Report of the 1983 NASA/ASEE Summer Faculty Workshop, NASA, 1985.

Johnson, T.J.R.: An Algorithm for the Resource-Constrained, Project Scheduling Problem. Unpublished Ph.D. Thesis, Massachusetts Institute of Technology, 1967.

Kelly, J.; Walker, M.: Critical-Path Planning and Scheduling. Proceedings of the Eastern Joint Computer Conference, 1959.

Kranzler, D. A.: An Integrated Stowage Logistics Clerk For Space Station Autonomy. Bachelor of Science Thesis for the Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, Massachusetts, June, 1986.

Kurtulus, I.; Davis, E.W.: Multi-Project Scheduling: Categorization of Heuristic Rules Performance. Management Science, Vol. 28, No. 2, February, 1982, pp. 161-172.

Larson, R.C.; Odoni, A.R.: Urban Operations Research. Prentiss-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

Lawler, E.L.; Lenstra, J.K.: Machine Scheduling with Precedence Constraints. Stichting Mathematisch Centrum, Amsterdam, Department of Operations Research, September, 1981.

Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G.: Recent Developments in Deterministic Sequencing and Scheduling: A Survey. Stichting Mathematisch Centrum, Amsterdam, Department of Operations Research, August, 1981.

Litsov, A.N.; Bulyko, V.I.: Principles of Organization of Rational Schedules for Crew Work and Rest During a Long-Term Spaceflight. USSR Report: Space Biology and Aerospace Medicine, Vol. 17, No. 4, July-August, 1983, pp. 9-13.

Mandeville, D.W.: The Development of Network Analysis Resource Balancing Methods From Assembly Line Balancing Techniques. Unpublished M.S. Thesis, Purdue University, Lafayette, Indiana, 1965.

Marsh, A.K. (1984a): NASA to Demonstrate Artificial Intelligence in Flight Operations. Aviation Week & Space Technology, September 17, 1984, Vol. 121, No. 12, p. 79.

Marsh, A.K. (1984b): Pace of Artificial Intelligence Research Shows Acceleration. Aviation Week & Space Technology, December 10, 1984, Vol. 121, No. 24, pp. 24-25.

Mathis, F.H.: Mathematical Programming Techniques for Scheduling Spacelab Crew Activities and Experiment Operations. NASA/ASEE 1981 Summer Faculty Research Fellowship Program. N82-17075, August 14, 1981.

Mogilensky, J.; Scarl, E.A.; Dalton, R.E.: Manned Spaceflight Activity Planning With Knowledge-Based Systems. AIAA Computers in Aerospace Conference, Hartford, CT, October, 1983.

Murphy, W.W.; Krusemark, K.A.; Moyer, R.W.: Increased Crew Activities Scheduling Effectiveness Through the Use of Computer Techniques. Human Factors, Vol. 10, No. 1, February, 1968, pp. 57-62.

NASA Space Human Factors Office: Space Station Operational Simulation (OpSim). NASA Ames Research Center, Space Human Factors Office, and Sterling Software, November 25, 1986.

Papadimitriou, C.H.; Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentiss-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

Pascoe, T.L.: An Experimental Comparison of Heuristic Methods for Allocating Resources. Unpublished Ph.D. Thesis, Cambridge University, 1965.

Patterson, J.H.: A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem. Management Science, Vol. 30, No. 7, July, 1984, pp. 854-867.

Patterson, J.H.: Project Scheduling: The Effect of Problem Structure on Heuristic Performance. Naval Research Logistics Quarterly, Vol. 23, March, 1976, pp. 95-123.

Patterson, J.H.: Alternate Methods Of Project Scheduling With Limited Resources. Naval Research Logistics Quarterly, Vol. 20, 1973, pp. 767-784.

Press, W.H.; Flannery, B.P.; Teukolsky, S.A.; Vetterling, W.T.: "Combinatorial Minimization: The Method of Simulated Annealing." Section 10.9 of Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, New York, New York, 1986, pp. 326-334.

Pritsker, A.A.B.; Watters, L.J.; Wolfe, P.M.: Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. Management Science, Vol. 16, No. 1, September, 1969, pp. 93-108.

Psaraftis, H.N.: K-Interchange Procedures for Local Search in a Precedence-Constrained Routing Problem. European Journal of Operational Research, Vol. 13, 1983, pp. 391-402.

Rook, F.W.; Odubiyi, J.B.: An Expert System for Satellite Orbit Control (ESSOC). Contel SPACECOM, Gaithersburg, Maryland, undated.

Scarl, E.A.; Engelman, C.; Pazzani, M.J.; Millen, J.: The KNOBS System. The MITRE Corporation, Bedford, Massachusetts, undated.

Slowinski, R.: Two Approaches to Problems of Resource Allocation Among Project Activities--A Comparative Study. Journal of the Operational Research Society, Vol. 31, No. 8, August, 1980, pp. 711-723.

Smith, S.F.: Exploiting Temporal Knowledge to Organize Constraints. Intelligent Systems Laboratory, Carnegie-Mellon University, Pittsburgh, PA, July 19, 1983.

Smith, S.F.; Ow, P.S.: The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks, Proceedings 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, August, 1985.

Stein, K.J.; et al: Boeing Accelerates Research, Dissemination of Technology. Aviation Week & Space Technology, February 17, 1986, Vol. 124, No. 7, pp. 71-79.

Stevens, L.: Window on Project Management: Is There a Mac in Your Plans? MacWEEK, September 22, 1987, pp. 21-24.

Stinson, J.P.; Davis, E.W.; Khumawala, B.M.: Multiple Resource--Constrained Scheduling Using Branch and Bound. AIEE Transactions, Vol. 10, No. 3, September, 1978, pp. 252-259.

STSC: APL*PLUS® System for the Macintosh, Reference Manual and User's Guide, STSC, Inc., Rockville, Maryland, 1986.

Talbot, F.B.; Patterson, J.H.: An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained-Scheduling Problems. Management Science, Vol. 24, No. 11, July 1978, pp. 1163-1174.

Touchton, B.: "SSES Fact Sheet." Technology Applications, Inc., Jacksonville, Florida.

Townsend, W.B.: Artificial Intelligence Techniques for Industrial Applications in Job Shop Scheduling. Naval Postgraduate School, Monterey California. M.S. Thesis, June, 1983. AD-A132-164.

Vere, S.A. (1983a): Planning in Time: Windows and Durations for Activities and Goals. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAM1-5, No. 3, May, 1983.

Vere, S.A. (1983b): Planning Spacecraft Activities with a Domain Independent Planner. AIAA Computers in Aerospace Conference, Hartford, CT, October, 1983.

Vere, S.A.: Deviser: an AI Planner for Spacecraft Operations. Aerospace America, Vol. 23, No. 4, April, 1985, pp. 50-53.

Wagner, R.E.: Expert System for Spacecraft Command and Control. AIAA Computers in Aerospace Conference, Hartford, CT, October, 1983, pp. 216-223.

Appendix A: Test Scheduling Problems

The following problems were used to test the algorithms described in Section 5. The first eight problems are all of the type known as resource constrained multi-project scheduling (see Section 4.3.1), and are all taken from previous studies of these problems. They are used to provide a comparison to previously published results. The rest of the problems are constructed to demonstrate algorithmic performance on problems containing constraints beyond the framework of traditional resource constrained multi-project scheduling.

Included in the description of each problem is a complete listing of each problem data including job parameters such as duration, resource usage, and time and target constraints. When crewmembers have different performance times for a job, each of these performance times is listed (note: resource constrained multi-project scheduling problems do not involve crewmembers, and therefore no individual performance times are given).

A.1 Summary Parameters

Following the statement of each problem is a listing of summary parameters which are computed from the problem data and which may be useful in describing problem complexity. All of these parameters are all taken from [Patterson, 1976], although some have been modified to encompass the more general format of the crew activity planning problem. Some of these were initially described in [Johnson, 1967; Pascoe, 1965; Davis, 1969; and Davis, 1973b]. The interested reader should refer to [Patterson, 1976] for further explanation of the importance of these parameters. These parameters are described as follows:

Number of Projects — This is the number of subsets of the jobs which contain no interrelating time constraints, i.e., a job is in a given project if, and only if, there is another job in that project to which it is related by a time constraint.

Number of Nodes — This is the total number of jobs. For problems taken from the resource constrained multi-project scheduling literature, dummy nodes (of 0 activity duration) have been included denoting the start and end of the scheduling interval.

Number of Arcs — This is the total number of non-redundant time constraints relating the jobs. Again, to maintain continuity with the resource constrained multi-project scheduling literature, dummy arcs have been included in these problems to denote precedence between jobs and the start and beginning of the scheduling interval.

Total Activity Density	For each activity j, compute T_j, which is defined as T_j = Max {0, Number of activities preceding activity j - Number of activities succeeding activity j}. Total Activity Density = $\sum_j T_j$
Average Activity Density	(Total Activity Density) ÷ Number of Nodes
Complexity	Number of Arcs ÷ Number of Nodes
Activity Duration, d_j	This is the time taken to complete Job j. For jobs in which different crewmembers have different performance times, d_j is the performance time of the crewmember which can complete it fastest. In the case of jobs which require several crewmembers, d_j is performance time of the subset of crewmembers which can complete it fastest. Sum of Activity Durations = $\sum_j d_j$ Average Activity Duration = $(\sum_j d_j)$ ÷ Number of Nodes Variance = $\frac{(d_j - \text{Average Activity Duration})^2}{\text{Number of Nodes} - 1}$ Maximum = $\max_j \{d_j\}$
Critical Path Length, CP_p	For each project p, the critical path length CP_p can be computed, as described in Section 4.2. The sum, average, and variance of the critical path lengths can then be computed in the same manner as they were for Activity Duration. For problems containing only one project, only the length of that single critical path is presented. Maximum Critical Path Length $CP_{max} = \max_p \{CP_p\}$
Total Slack (Float) of Job j, TF_j	Subsequent to critical path analysis, the Total Slack of Job j is defined as the difference between the job's latest start time and its earliest start time based on a schedule which completes each project within its critical path duration, ignoring all resource constraints. For example, all jobs which are part of a project's critical path have zero Total Float.

$$\text{Total Slack} = \sum_j TF_j$$

$$\text{\# of Activities With Positive Slack} = \sum_j \begin{cases} 1 \text{ if } TF_j > 0 \\ 0 \text{ if } TF_j = 0 \end{cases}$$

$$\text{Average Total Slack Per Activity} = \frac{\text{Total Slack}}{\text{Number of Nodes}}$$

$$\text{Project Density - Total} = \frac{\sum_j d_j}{\sum_j d_j + \sum_j TF_j}$$

$$\text{Total Slack Ratio} = (\sum_j TF_j) \div CP_{max}$$

$$\text{Average Slack Ratio} = \text{Average Total Slack} \div CP_{max}$$

Free Slack (Float) of Job j, FF_j

The Free Slack of a job is defined as the difference between the latest and earliest start time of each job, given that any jobs which follow it start at their earliest possible start time. The Free Slack of a job is alway less than or equal to the Total Slack.

$$\text{Total Free Slack} = \sum_j FF_j$$

$$\text{\# of Activities With Positive Free Slack} = \sum_j \begin{cases} 1 \text{ if } FF_j > 0 \\ 0 \text{ if } FF_j = 0 \end{cases}$$

$$\text{Average Free Slack Per Activity} = \frac{\text{Total Free Slack}}{\text{Number of Nodes}}$$

$$\text{Project Density - Free} = \frac{\sum_j d_j}{\sum_j d_j + \sum_j FF_j}$$

Percent of Demand, $PCTR_k$

This is the percent of jobs which require positive amounts of Resource k. Let the amount of Resource k used by Job j be denoted as r_{jk}, and the resource limit of Resource k be R_k.

$$PCTR_k = \frac{\sum_j \begin{cases} 1 \text{ if } r_{jk} > 0 \\ 0 \text{ if } r_{jk} = 0 \end{cases}}{\text{Number of Nodes}}$$

Utilization of Resource k, $UTIL_k$

$$UTIL_k = \frac{\sum_j r_{jk} \cdot d_j}{R_k \cdot CP_{max}}$$

Avg Qty When Demanded, $DMND_k$ This is the average of Resource k demanded when it is required by an activity.

$$DMND_k = \frac{\sum_j r_{jk}}{\sum_j \begin{cases} 1 \text{ if } r_{jk} > 0 \\ 0 \text{ if } r_{jk} = 0 \end{cases}}$$

Resource Constrainedness $= DMND_k \div R_k$

Resource Constrainedness Over Time, $TCON_k$

$$TCON_k = \frac{\sum_j r_{jk} \cdot d_j}{R_k \cdot CP_{max} \cdot \sum \begin{cases} 1 \text{ if } r_{jk} > 0 \\ 0 \text{ if } r_{jk} = 0 \end{cases}}$$

Resource Constrainedness (using All Activities as a base), $ACON_k$

$$ACON_k = \frac{\sum_j r_{jk}}{R_k \cdot \text{Number of Nodes}}$$

Obstruction Factor, OFACTk The Obstruction Factor of Resource k is defined as

$$OFACT_k = \frac{\text{"Excess" Resource Requirement}_k}{\text{Resource Work Content}_k}$$

In order to compute the Excess Resource Requirement of Resource k it is first necessary to generate a schedule where each job is scheduled at its earliest possible start time (ignoring any resource limits). The Excess Resource Requirement of Resource k is then defined as:

$\sum \max \{0, (\text{Demand For Resource k}) - R_k\}$

where the demand for Resource k is based on this all early start schedule.

The Resource Work Content of Resource k is = $\sum_j r_{jk} \cdot d_j$

Underutilization Factor, $UFACT_k$

$$UFACT_k = \frac{\sum \max\{0, R_k - Demand_k\}}{\text{Resource Work Content}_k}$$

Where the demand for Resource k is based on an all early start schedule (See Obstruction Factor, above).

Excess Demand Time Periods, $NOVER_k$ This parameter describes the total amount of time for which the demand for Resource k exceeds the availability of Resource k, based on an all early start schedule.

$$NOVER_k = \sum \begin{cases} 1 \text{ if } Demand_k > R_k \\ 0 \text{ if } Demand_k \leq R_k \end{cases}$$

Time Underutilization, $NUNDER_k$ This parameter describes the total amount of time for which the availability of Resource k exceeds or equals the availability of Resource k, based on an all early start schedule.

$$NUNDER_k = \sum \begin{cases} 1 \text{ if } R_k \geq Demand_k \\ 0 \text{ if } R_k < Demand_k \end{cases}$$

After the listing of summary parameters is a description of an optimal (or best known) solution to the problem, based upon the criteria of minimizing the completion time of the last job.

Finally, results from the testing of the various algorithms are presented, along with graphs showing comparisons between the performances of the ten heuristics (and their randomized versions), on that particular problem (see Section 5.2). Comparison of results between the different problems were presented in Section 5.3.

A.2 Test Problems 1 and 2

These problems are from [Mandeville, 1965] and were used by [Davis, 1969, p. 198]. These two single-resource problems are identical except for different usages of the resource by the jobs and a different resource usage limit.

Table A.1: Problem Data for Problems 1 and 2

Job	Duration	Resource Usage (Problem 1)	Resource Usage (Problem 2)
1	3	2	2
2	3	3	2
3	1	2	2
4	2	3	3
5	2	4	3
6	3	2	1
7	3	1	1

Resource Limit, Problem 1 = 5
Resource Limit, Problem 2 = 4

The following graph shows the precedence relations between the jobs. Nodes S and T are dummy nodes denoting the start and end of the schedule.

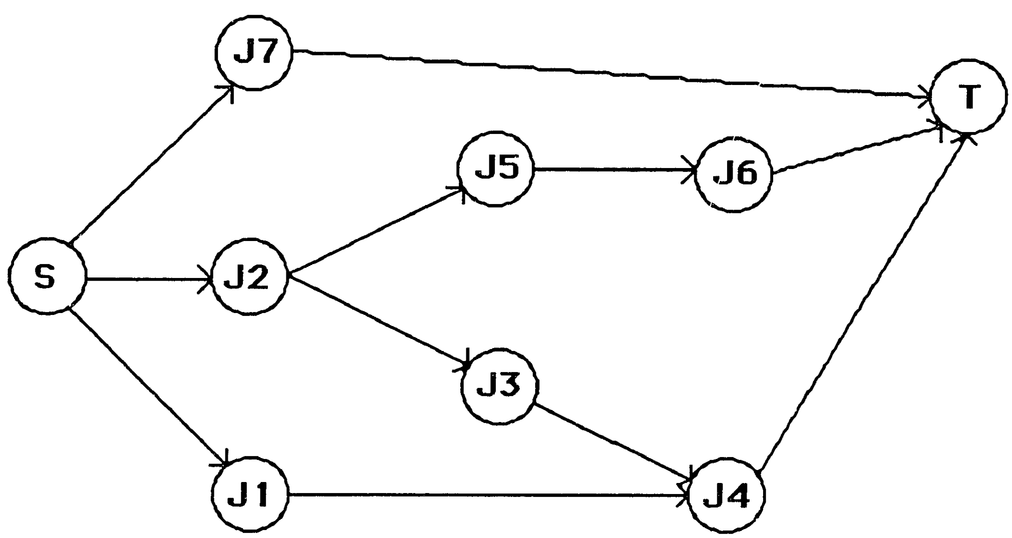


Figure A.1: Precedence Network, Problems 1 and 2

Problems 1 and 2: Summary Statistics

NUMBER OF PROJECTS	= 1
NUMBER OF NODES	= 9 INCLUDING 2 DUMMY NODES
NUMBER OF ARCS	= 11 INCLUDING 6 DUMMY ARCS
TOTAL ACTIVITY DENSITY	= 5
AVERAGE ACTIVITY DENSITY	= 0.56
COMPLEXITY	= 1.22

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 17
	Average	= 1.89
	Variance	= 0.72
	Maximum	= 3
CRITICAL PATH LENGTH	Sum	= 8
ACTIVITY SLACK	Total Slack	= 12
	# Of Activities With Positive Slack	= 4 (44.44%)
	Average Slack Per Activity	= 1.33
	Project Density - Total	= 0.59
	Total Slack Ratio	= 1.50
	Average Slack Ratio	= 0.17
FREE SLACK	Total	= 8
	# Of Activities With Positive Free Slack	= 3 (33.33%)
	Average Free Slack Per Activity	= 0.89
	Project Density - Free	= 0.68

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	NO
Audio/Vibration Resource?	NO
Number Of Other Resources	= 1

	PROBLEM 1	PROBLEM 2
PERCENT OF DEMAND	Resource 1 = 0.78	Resource 1 = 0.78
RESOURCE UTILIZATION	Resource 1 = 1.00	Resource 1 = 1.00
AVG QTY WHEN DEMANDED	Resource 1 = 2.43	Resource 1 = 2.00
RESOURCE CONSTRAINEDNESS	Resource 1 = 0.49	Resource 1 = 0.50
RES CONSTRAINEDNESS OVER TIME	Resource 1 = 0.14	Resource 1 = 0.14
RES CONSTR (ALL ACTIVITIES)	Resource 1 = 0.38	Resource 1 = 0.39
OBSTRUCTION FACTOR	Resource 1 = 0.1500	Resource 1 = 0.1875
UNDERUTILIZATION FACTOR	Resource 1 = 0.1500	Resource 1 = 0.1875
EXCESS DEMAND TIME PERIODS	Resource 1 = 5.00	Resource 1 = 5.00
TIME UNDERUTILIZATION	Resource 1 = 3.00	Resource 1 = 3.00

Table A.2: Optimal Solution, Problems 1 and 2

The following solution of duration 8 is optimal for both Problem 1 and Problem 2.

Job	Start Time	End Time
1	0	3
2	0	3
3	5	6
4	6	8
5	3	5
6	5	8
7	3	6

For Problem 1, this produces a schedule with a constant resource usage of 5.
For Problem 2, this produces a schedule with a constant resource usage of 4.

Job ordering to dispatcher which produces optimal schedule = 2, 5, 1, 3, 4, 6, 7.

Results

Table A.3: Quality of Initial Solutions, Problems 1 and 2

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average) Problem 1	Problem 2	Standard Deviation in Rand. Initial Solution Problem 1	Problem 2
1 SJF	11	10.83	10.80	1.631	1.738
2 LJF	8	10.04	10.46	1.536	1.694
3 MSLK	8	10.70	11.04	1.646	1.749
4 GRD	8	9.37	9.78	1.629	1.604
5 GRR	8	9.36	9.27	1.453	1.355
6 JER	11	10.78	10.81	1.706	1.419
7 LFT	8	10.51	10.85	1.769	1.492
8 EST	9	10.41	10.88	1.828	1.627
9 MCM	9	10.28	9.91	1.839	1.668
10 CCM	9	10.00	10.14	1.811	1.625
Average	8.9	10.23	10.39	1.675	1.602

The randomized initial solutions are based on the average of 100 single iteration scheduling attempts. Figures A.2 and A.3 show graphical comparisons of the deterministic initial solution and the average randomized initial solution. For this problem, the number of possible alternatives for selecting job ordering was small (182), so that it was possible to form schedules with each of these alternatives and thus actually compute the true value for the expected randomized initial solution using Heuristic 6. For both Problems 1 and 2, this expected value turned out to be equal to 10.712 (with a standard deviation of 1.686). This is close to the values actually obtained.

Figure A.2: Initial Solutions, Problem 1

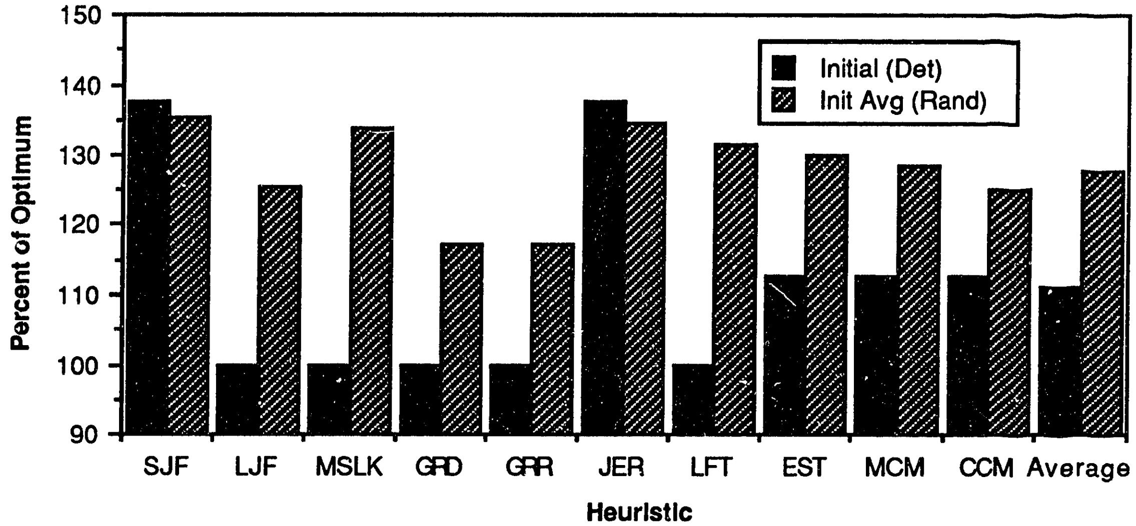


Figure A.3: Initial Solutions, Problem 2

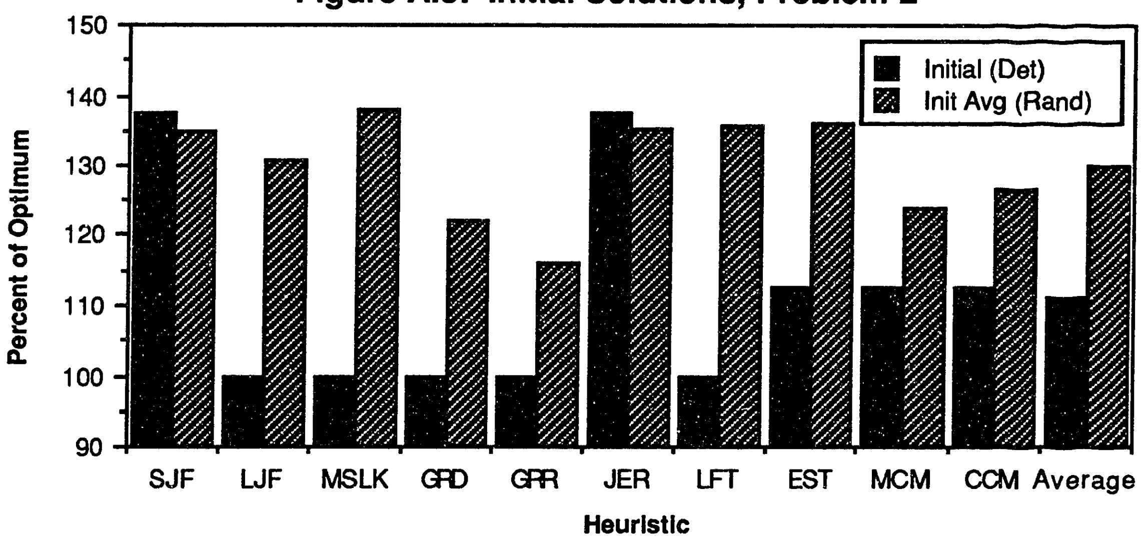


Table A.4: Average Number of Iterations to Optimal Solution, Problems 1 and 2

Heuristic	Problem 1 (See Figure A.4) Deterministic	Problem 1 (See Figure A.4) Randomized	Problem 2 (See Figure A.5) Deterministic	Problem 2 (See Figure A.5) Randomized
1 SJF	3.333	6.000	3.333	1.667
2 LJF	1.000	5.000	1.000	3.667
3 MSLK	1.000	5.667	1.000	3.333
4 GRD	1.000	1.000	1.000	2.000
5 GRR	1.000	2.333	1.000	1.667
6 JER	2.000	5.667	2.000	3.000
7 LFT	1.000	2.667	1.000	4.000
8 EST	2.000	3.667	2.000	2.667
9 MCM	8.667	5.667	14.667	3.333
10 CCM	10.333	3.000	9.333	3.000
Average	3.133	4.067	3.633	2.833

The above data is averaged over 3 scheduling attempts for each heuristic. From these results it is seen that (with the exceptions of Heuristics 9 and 10) it is usually better to use the deterministic algorithm than the randomized one (for this problem). This is not unexpected with a problem that is this trivial, because while the deterministic search moves directly towards the optimum, the randomized search tends to get sidetracked. For this particular problem, Heuristics 9 and 10 send the algorithm searching very strongly in the wrong direction. Hence it is desirable that the algorithm gets sidetracked by the randomization (i.e.,there is a probability that the algorithm will look in a directions other than the one indicated by the deterministic heuristic).

Figure A.4: Iterations to Optimum, Problem 1

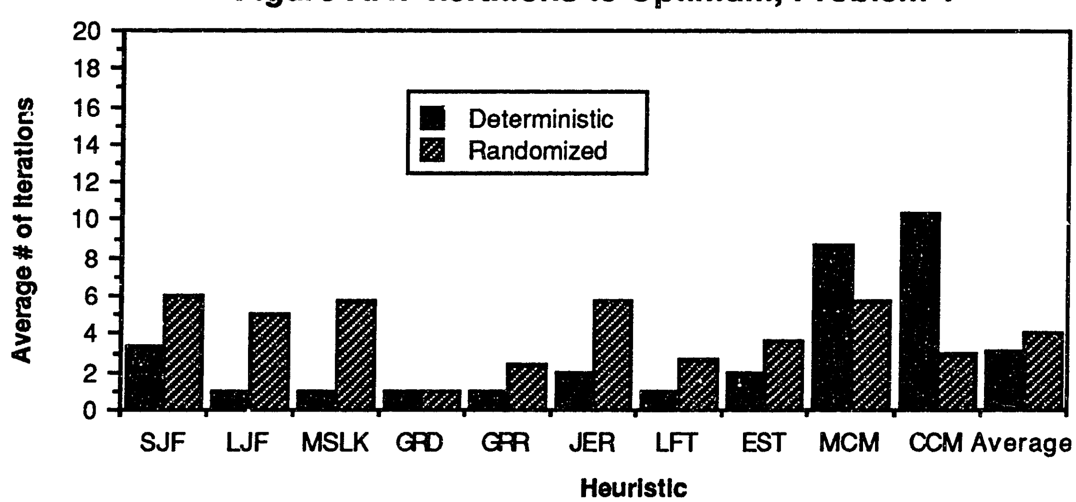
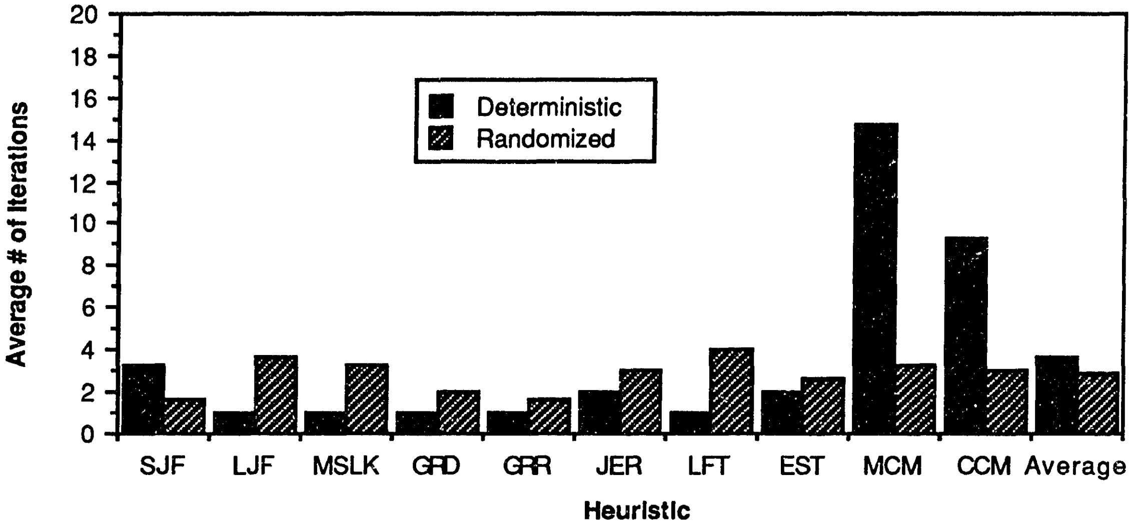


Figure A.5: Iterations to Optimum, Problem 2



A.3 Test Problem 3

This problem is from [Brand, Shaffer, and Meyer, 1964] and was used by [Davis, 1969, p. 199].

Table A.5: Problem Data for Problem 3

Job	Duration	Resource Usage (Resource 1)	Resource Usage (Resource 2)	Resource Usage Resource 3
1	12	1	0	0
2	8	0	0	0
3	7	0	0	0
4	3	0	0	0
5	12	1	0	1
6	5	1	0	0
7	2	0	0	0
8	9	0	1	1
9	7	0	0	1
10	4	0	0	1
11	6	0	1	0
12	10	0	0	0
Resource Limits		2	1	2

The following graph shows the precedence relations between the jobs. Nodes S and T are dummy nodes denoting the start and end of the schedule.

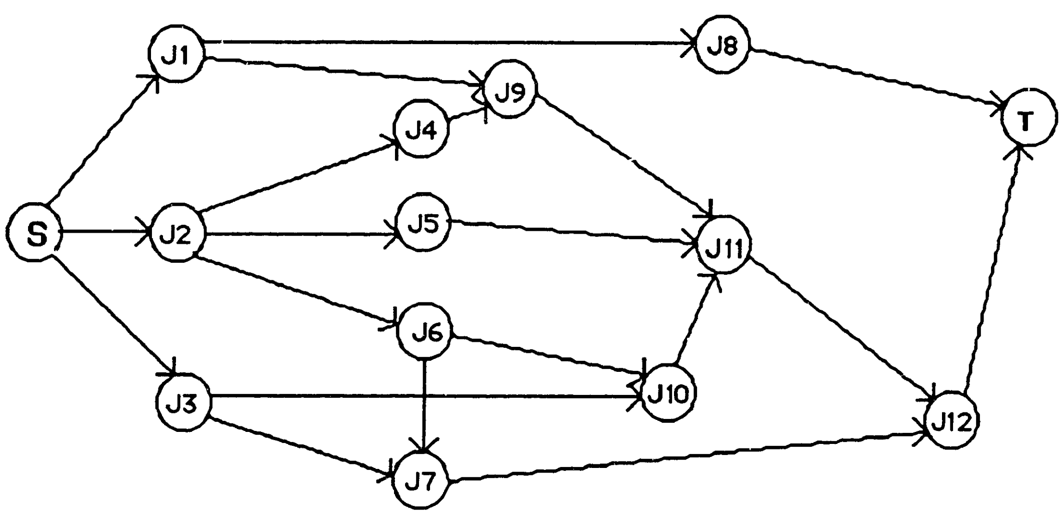


Figure A.6: Precedence Network, Problem 3

Problem 3: Summary Statistics

NUMBER OF PROJECTS	= 1
NUMBER OF NODES	= 14 INCLUDING 2 DUMMY NODES
NUMBER OF ARCS	= 20 INCLUDING 5 DUMMY ARCS
TOTAL ACTIVITY DENSITY	= 22
AVERAGE ACTIVITY DENSITY	= 1.57
COMPLEXITY	= 1.43

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 85
	Average	= 6.07
	Variance	= 10.09
	Maximum	= 12
CRITICAL PATH LENGTH	Sum	= 36
ACTIVITY SLACK	Total Slack	= 45
	# Of Activities With Positive Slack	= 8 (57.14%)
	Average Slack Per Activity	= 3.21
	Project Density - Total	= 0.65
	Total Slack Ratio	= 1.25
	Average Slack Ratio	= 0.09
FREE SLACK	Total	= 37
	# Of Activities With Positive Free Slack	= 6 (42.86%)
	Average Free Slack Per Activity	= 2.64
	Project Density - Free	= 0.70

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	NO
Audio/Vibration Resource?	NO
Number Of Other Resources	= 3

PERCENT OF DEMAND	Resource 1	= 0.21
	Resource 2	= 0.14
	Resource 3	= 0.29
	AVERAGE	= 0.21
	VARIANCE	= 0.0051
RESOURCE UTILIZATION	Resource 1	= 0.40
	Resource 2	= 0.42
	Resource 3	= 0.44
	AVERAGE	= 0.42
	VARIANCE	= 0.0005
AVG QTY WHEN DEMANDED	Resource 1	= 1.00
	Resource 2	= 1.00
	Resource 3	= 1.00
	AVERAGE	= 1.00
	VARIANCE	= 0.0000
RESOURCE CONSTRAINEDNESS	Resource 1	= 0.50
	Resource 2	= 1.00
	Resource 3	= 0.50
	AVERAGE	= 0.67
	VARIANCE	= 0.0833

RES CONSTRAINEDNESS OVER TIME	Resource 1 =	0.13
	Resource 2 =	0.21
	Resource 3 =	0.11
	AVERAGE =	0.15
	VARIANCE =	0.0026
RES CONSTR (ALL ACTIVITIES)	Resource 1 =	0.11
	Resource 2 =	0.14
	Resource 3 =	0.14
	AVERAGE =	0.13
	VARIANCE =	0.0004
OBSTRUCTION FACTOR	Resource 1 =	0.1379
	Resource 2 =	0.0667
	Resource 3 =	0.3438
	TOTAL =	0.5483
UNDERUTILIZATION FACTOR	Resource 1 =	1.6207
	Resource 2 =	1.4667
	Resource 3 =	1.5938
	TOTAL =	4.6811
EXCESS DEMAND TIME PERIODS	Resource 1 =	4.00
	Resource 2 =	1.00
	Resource 3 =	7.00
	AVERAGE =	4.00
	VARIANCE =	9.0000
TIME UNDERUTILIZATION	Resource 1 =	32.00
	Resource 2 =	35.00
	Resource 3 =	29.00
	AVERAGE =	32.00
	VARIANCE =	9.0000

Table A.6: Optimal Solution, Problem 3

The following solution is optimal, with a duration of 39.

Job	Start Time	End Time
1	0	12
2	0	8
3	0	7
4	8	11
5	8	20
6	12	17
7	17	19
8	29	38
9	12	19
10	19	23
11	23	29
12	29	39

Resource 1 Levels: 1 from 0 to 8; 2 from 8 to 17; 1 from 17 to 20; 0 from 20 to 39.

Resource 2 Levels: 0 from 0 to 23; 1 from 23 to 38; 0 from 38 to 39.

Resource 3 Levels: 0 from 0 to 8; 1 from 8 to 12; 2 from 12 to 20; 1 from 20 to 23; 0 from 23 to 29; 1 from 29 to 38; 0 from 38 to 39.

Job ordering to dispatcher which produces optimal schedule = 2, 3, 1, 5, 4, 9, 6, 10, 7, 11, 12, 8.

Results

Table A.7: Quality of Initial Solution, Problem 3

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average)	Standard Deviation in Rand. Initial Solution
1 SJF	47	46.78	2.941
2 LJF	44	45.35	2.523
3 MSLK	39	46.28	2.657
4 GRD	44	44.51	1.127
5 GRR	43	44.64	1.769
6 JER	43	45.92	2.564
7 LFT	40	45.36	2.678
8 EST	43	44.76	2.093
9 MCM	46	45.19	1.759
10 CCM	46	45.46	1.977
Average	43.5	45.425	2.272

The randomized initial solutions are based on the average of 100 single iteration scheduling attempts. Figure A.7 shows a graphical comparison of the deterministic initial solution and the average randomized initial solution.

Figure A.7: Initial Solutions, Problem 3

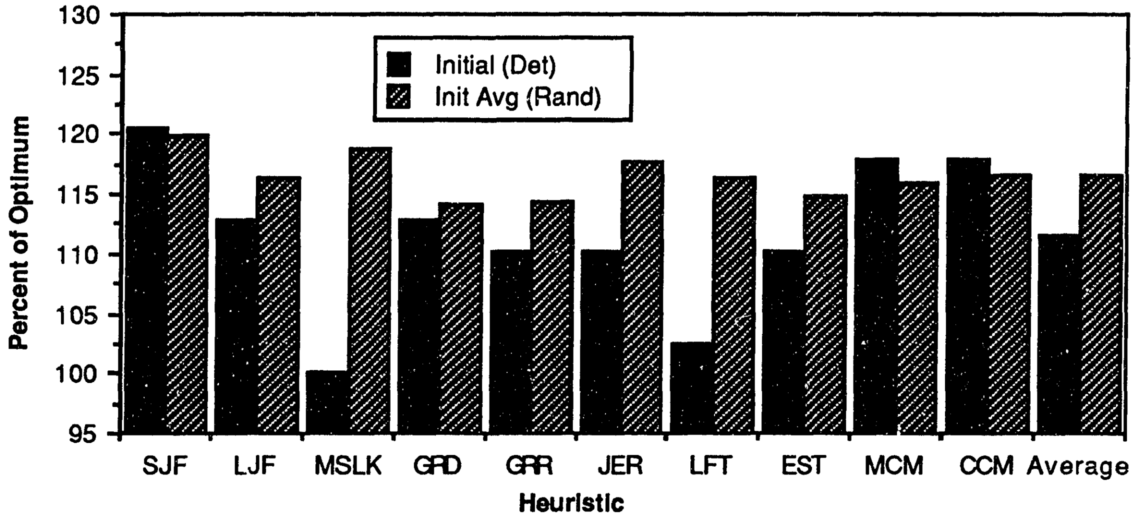


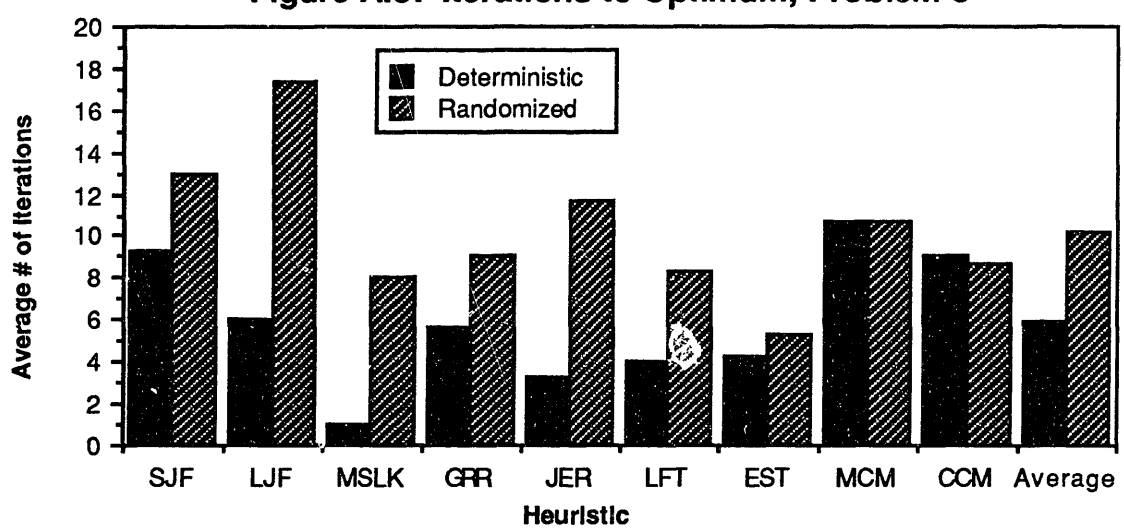
Table A.8: Average Number of Iterations to Optimal Solution, Problem 3

	Problem 3 (See Figure A.8)	
Heuristic	Deterministic	Randomized
1 SJF	9.333	13.000
2 LJF	6.000	17.333
3 MSLK	1.000	8.000
4 GRD	*	*
5 GRR	5.667	9.000
6 JER	3.333	11.667
7 LFT	4.000	8.333
8 EST	4.333	5.333
9 MCM	10.667	10.667
10 CCM	9.000	8.667
Average	5.926	10.222

The above data is averaged over 3 scheduling attempts for each heuristic. No results are shown for Heuristic 4 because the Greatest Resource Demand Heuristic failed to converge for this problem. This occurred because, when starting the algorithm, Jobs 1, 2, and 3 are pending. However, Jobs 2 and 3 use no resources, so they get a zero weighting from the algorithm. As the algorithm progresses from iteration to iteration, the priorities of Jobs 2 and 3 are increased, but their total rating remains zero, because their total rating is computed as their heuristic weight multiplied by their priority. As a result, Jobs 2 and 3 always end up being scheduled after Job 1, even though this may not be desirable to form an optimal schedule.

One possible way to fix this "bug" in the algorithm would be to add some small number to the weight of each job produced by the heuristic. The iterative algorithm would thus eventually succeed in boosting the rating of these zero weight jobs above the ratings of other jobs, if it believes that a better solution would result.

Figure A.8: Iterations to Optimum, Problem 3



A.4 Test Problem 4

This problem is from [Davis, 1969, p. 148].

Table A.9: Problem Data for Problem 4

Job	Duration	Resource Usage (Resource 1)	Resource Usage (Resource 2)	Resource Usage Resource 3
1	3	3	5	2
2	2	5	4	3
3	3	5	2	2
4	8	4	1	4
5	4	5	5	4
6	4	3	5	2
7	1	2	4	4
8	5	3	2	2
9	2	3	2	4
10	5	3	3	2
11	4	4	1	4
12	1	1	4	4
13	1	2	2	2
14	1	5	5	4
15	3	1	5	4
16	7	4	5	4
17	5	3	2	3
18	3	5	3	3
19	4	2	4	6
20	4	1	6	2
21	4	3	2	1
22	3	1	1	4
23	2	2	2	1
24	1	1	1	3
25	2	2	2	2
Resource Limits		8	10	10

The following graph shows the precedence relations between the jobs. Nodes S and T are dummy nodes denoting the start and end of the schedule.

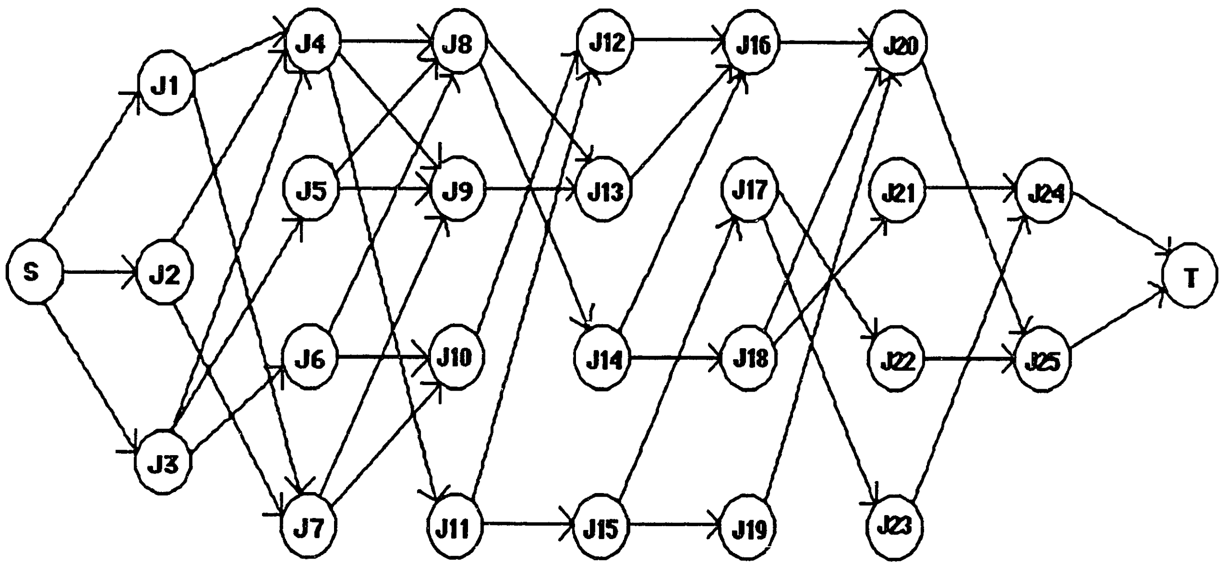


Figure A.9: Precedence Network, Problem 4

Problem 4: Summary Statistics

NUMBER OF PROJECTS	= 1
NUMBER OF NODES	= 27 INCLUDING 2 DUMMY NODES
NUMBER OF ARCS	= 43 INCLUDING 5 DUMMY ARCS
TOTAL ACTIVITY DENSITY	= 109
AVERAGE ACTIVITY DENSITY	= 4.04
COMPLEXITY	= 1.59

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 82
	Average	= 3.04
	Variance	= 3.17
	Maximum	= 8
CRITICAL PATH LENGTH	Sum	= 30
ACTIVITY SLACK	Total Slack	= 50
	# Of Activities With Positive Slack	= 16 (59.26%)
	Average Slack Per Activity	= 1.85
	Project Density - Total	= 0.62
	Total Slack Ratio	= 1.67
	Average Slack Ratio	= 0.06
FREE SLACK	Total	= 24
	# Of Activities With Positive Free Slack	= 10 (37.04%)
	Average Free Slack Per Activity	= 0.89
	Project Density - Free	= 0.77

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	NO
Audio/Vibration Resource?	NO
Number Of Other Resources	= 3

PERCENT OF DEMAND	Resource 1	= 0.93
	Resource 2	= 0.93
	Resource 3	= 0.93
	AVERAGE	= 0.93
	VARIANCE	= 0.0000
RESOURCE UTILIZATION	Resource 1	= 1.07
	Resource 2	= 0.85
	Resource 3	= 0.84
	AVERAGE	= 0.92
	VARIANCE	= 0.0170
AVG QTY WHEN DEMANDED	Resource 1	= 2.92
	Resource 2	= 3.12
	Resource 3	= 3.04
	AVERAGE	= 3.03
	VARIANCE	= 0.0101
RESOURCE CONSTRAINEDNESS	Resource 1	= 0.36
	Resource 2	= 0.31
	Resource 3	= 0.30
	AVERAGE	= 0.33
	VARIANCE	= 0.0011

RES CONSTRAINEDNESS OVER TIME	Resource 1	= 0.04
	Resource 2	= 0.03
	Resource 3	= 0.03
	AVERAGE	= 0.04
	VARIANCE	= 0.0000
RES CONSTR (ALL ACTIVITIES)	Resource 1	= 0.34
	Resource 2	= 0.29
	Resource 3	= 0.28
	AVERAGE	= 0.30
	VARIANCE	= 0.0009
OBSTRUCTION FACTOR	Resource 1	= 0.2374
	Resource 2	= 0.1181
	Resource 3	= 0.1067
	TOTAL	= 0.4622
UNDERUTILIZATION FACTOR	Resource 1	= 0.1712
	Resource 2	= 0.2992
	Resource 3	= 0.2925
	TOTAL	= 0.7629
EXCESS DEMAND TIME PERIODS	Resource 1	= 15.00
	Resource 2	= 13.00
	Resource 3	= 7.00
	AVERAGE	= 11.67
	VARIANCE	= 17.3333
TIME UNDERUTILIZATION	Resource 1	= 15.00
	Resource 2	= 17.00
	Resource 3	= 23.00
	AVERAGE	= 18.33
	VARIANCE	= 17.3333

Table A.10: Optimal Solution, Problem 4

The following solution is optimal, with a duration of 40.

Job	Start Time	End Time	Job	Start Time	End Time
1	0	3	16	26	33
2	7	9	17	30	35
3	0	3	18	23	26
4	9	17	19	26	30
5	3	7	20	33	37
6	3	7	21	33	37
7	9	10	22	35	38
8	17	22	23	37	39
9	21	23	24	39	40
10	10	15	25	38	40
11	17	21			
12	21	22			
13	23	24			
14	22	23			
15	23	26			

Resource Level vs. Time for Optimal Solution:

Time	Res. 1	Res. 2	Res. 3
0	8	7	4
3	8	10	6
7	5	4	3
9	6	5	8
10	7	4	6
15	4	1	4
17	7	3	6
21	7	8	10
22	8	7	8
23	8	10	9
24	6	8	7
26	6	9	10
30	7	7	7
33	7	10	6
35	5	9	7
37	3	3	5
38	4	4	3
39	3	3	5
40	0	0	0

Job ordering to dispatcher which produces optimal schedule =

3, 5, 2, 1, 4, 11, 6, 8, 14, 18, 7, 9, 10, 15, 13, 12, 16, 19, 20, 17, 22, 21, 25, 23, 24.

Results

Table A.11: Quality of Initial Solution, Problem 4

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average)	Standard Deviation in Rand. Initial Solution
SJF	48	47.32	3.026
LJF	46	46.45	1.438
MSLK	50	46.09	2.534
GRD	47	46.56	1.451
GRR	42	44.56	2.056
JER	42	46.58	2.178
LFT	41	46.27	2.315
EST	42	45.88	2.290
MCM	45	45.04	1.766
CCM	43	44.9	1.895
Average	44.6	45.965	2.146

The randomized initial solutions are based on the average of 100 single iteration scheduling attempts. Figure A.10 shows a graphical comparison of the deterministic initial solution and the average randomized initial solution.

Figure A.10: Initial Solutions, Problem 4

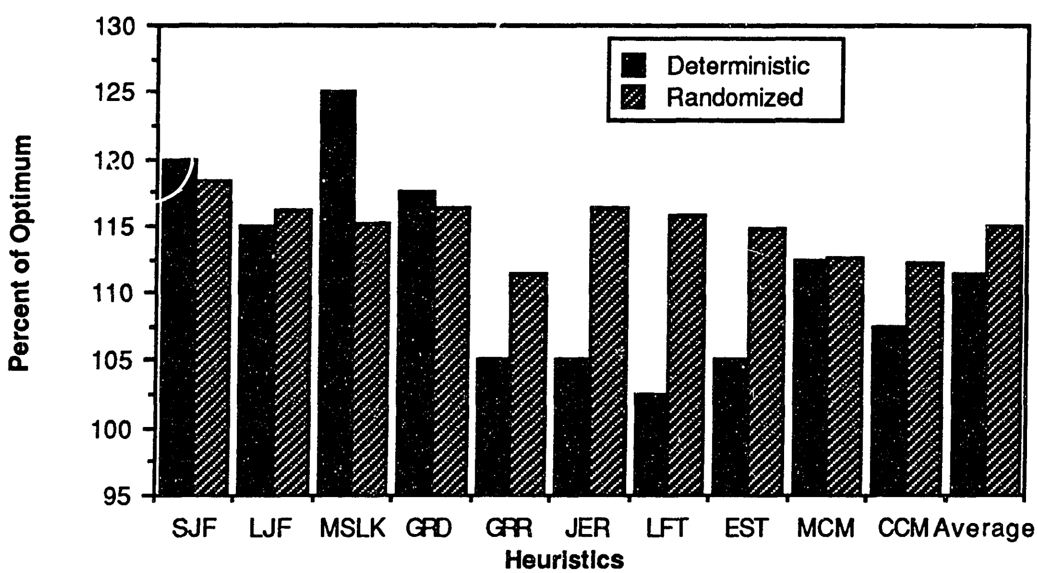


Table A.12: Average Number of Iterations to Optimal Solution, Problem 4

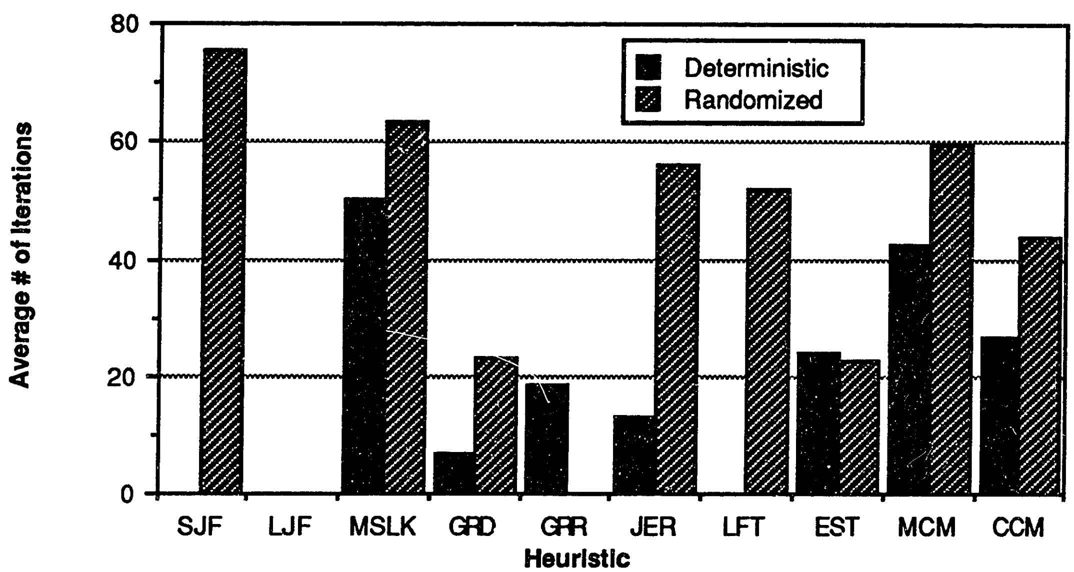
Heuristic	Problem 4 (See Figure A.11) Deterministic	Randomized
1 SJF	*	75.667
2 LJF	*	*
3 MSLK	50	63.333
4 GRD	7	23
5 GRR	18.333	*
6 JER	13.333	56
7 LFT	*	52
8 EST	24	22.667
9 MCM	42.333	59.667
10 CCM	26.667	43.667

The above data is averaged over 3 scheduling attempts for each heuristic. No results are shown for some of the heuristics because the iterative algorithm did not converge to optimality in 100 iterations for at least 1 of the 3 scheduling attempts.

Of the 30 deterministic scheduling attempts (10 heuristics x 3 attempts per heuristic), 23 had converged to optimality after 50 iterations, and 26 had converged to optimality after 100 iterations.

Of the 30 randomized scheduling attempts, 18 had converged to optimality after 50 iterations, and 28 had converged to optimality after 100 iterations.

Figure A.11: Iterations to Optimum, Problem 4



A.5 Test Problem 5

This problem is from [Davis, 1969, p. 200].

Table A.13: Problem Data for Problem 5

Job	Duration	Resource Usage (Resource 1)	Resource Usage (Resource 2)	Resource Usage Resource 3
1	3	3	5	2
2	4	5	4	3
3	3	5	2	2
4	4	4	1	4
5	3	5	5	4
6	6	3	5	2
7	5	2	4	4
8	2	3	2	2
9	2	3	2	4
10	4	3	3	2
11	8	4	1	4
12	3	1	4	4
13	2	2	2	2
14	3	5	5	4
15	2	1	5	4
16	3	4	5	4
17	2	3	2	3
18	5	5	3	3
19	3	2	4	6
20	2	1	6	2
21	5	3	2	1
22	3	1	1	4
23	2	2	2	1
24	3	1	1	3
25	3	2	2	2
Resource Limits		10	10	10

Note: The resource usages and limits are exactly the same as for Problem 4. Two typographical errors from [Davis, 1969] in the resource usages for Jobs 22 and 24 have been corrected.

The following graph shows the precedence relations between the jobs. Nodes S and T are dummy nodes denoting the start and end of the schedule.

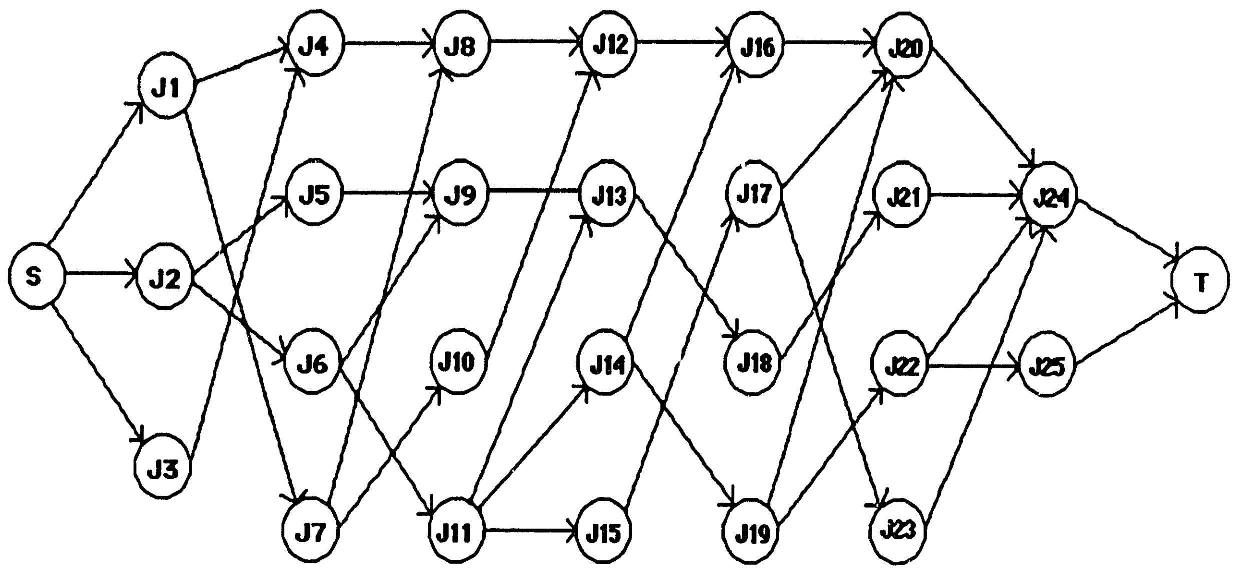


Figure A.12: Precedence Network, Problem 5

Problem 5: Summary Statistics

NUMBER OF PROJECTS	= 1
NUMBER OF NODES	= 27 INCLUDING 2 DUMMY NODES
NUMBER OF ARCS	= 38 INCLUDING 5 DUMMY ARCS
TOTAL ACTIVITY DENSITY	= 75
AVERAGE ACTIVITY DENSITY	= 2.78
COMPLEXITY	= 1.41

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 85
	Average	= 3.15
	Variance	= 2.06
	Maximum	= 8
CRITICAL PATH LENGTH	Sum	= 33
ACTIVITY SLACK	Total Slack	= 131
	# Of Activities With Positive Slack	= 18 (66.67%)
	Average Slack Per Activity	= 4.85
	Project Density - Total	= 0.39
	Total Slack Ratio	= 3.97
	Average Slack Ratio	= 0.15
FREE SLACK	Total	= 31
	# Of Activities With Positive Free Slack	= 8 (29.63%)
	Average Free Slack Per Activity	= 1.15
	Project Density - Free	= 0.73

STATISTICS RELATING TO THE RESOURCES

Crew Resource?		NO
Audio/Vibration Resource?		NO
Number Of Other Resources		= 3
PERCENT OF DEMAND	Resource 1	= 0.93
	Resource 2	= 0.93
	Resource 3	= 0.93
	AVERAGE	= 0.93
	VARIANCE	= 0.0000
RESOURCE UTILIZATION	Resource 1	= 0.80
	Resource 2	= 0.78
	Resource 3	= 0.79
	AVERAGE	= 0.79
	VARIANCE	= 0.0001
AVG QTY WHEN DEMANDED	Resource 1	= 2.92
	Resource 2	= 3.12
	Resource 3	= 3.04
	AVERAGE	= 3.03
	VARIANCE	= 0.0101
RESOURCE CONSTRAINEDNESS	Resource 1	= 0.29
	Resource 2	= 0.31
	Resource 3	= 0.30
	AVERAGE	= 0.30
	VARIANCE	= 0.0001

RES CONSTRAINEDNESS OVER TIME	Resource 1	= 0.03
	Resource 2	= 0.03
	Resource 3	= 0.03
	AVERAGE	= 0.03
	VARIANCE	= 0.0000
RES CONSTR (ALL ACTIVITIES)	Resource 1	= 0.27
	Resource 2	= 0.29
	Resource 3	= 0.28
	AVERAGE	= 0.28
	VARIANCE	= 0.0001
OBSTRUCTION FACTOR	Resource 1	= 0.1321
	Resource 2	= 0.1313
	Resource 3	= 0.1034
	TOTAL	= 0.3668
UNDERUTILIZATION FACTOR	Resource 1	= 0.3774
	Resource 2	= 0.4054
	Resource 3	= 0.3678
	TOTAL	= 1.1506
EXCESS DEMAND TIME PERIODS	Resource 1	= 11.00
	Resource 2	= 11.00
	Resource 3	= 7.00
	AVERAGE	= 9.67
	VARIANCE	= 5.3333
TIME UNDERUTILIZATION	Resource 1	= 22.00
	Resource 2	= 22.00
	Resource 3	= 26.00
	AVERAGE	= 23.33
	VARIANCE	= 5.3333

Table A.14: Optimal Solution, Problem 5

The following solution is optimal, with a duration of 35.

Job	Start Time	End Time	Job	Start Time	End Time
1	3	6	16	27	30
2	0	4	17	24	26
3	0	3	18	22	27
4	6	10	19	21	24
5	11	14	20	30	32
6	4	10	21	27	32
7	6	11	22	27	30
8	16	18	23	26	28
9	14	16	24	32	35
10	14	18	25	30	33
11	10	18			
12	24	27			
13	20	22			
14	18	21			
15	18	20			

Resource Level vs. Time for Optimal Solution:

Time	Res. 1	Res. 2	Res. 3
0	10	6	5
3	8	9	5
4	6	10	4
6	9	10	10
10	6	5	8
11	9	6	8
14	10	6	10
16	10	6	8
18	6	10	8
20	7	7	6
21	4	6	8
22	7	7	9
24	9	9	10
26	8	9	8
27	10	10	10
28	8	8	9
30	6	10	5
32	3	3	5
33	1	1	3
35	0	0	0

Job ordering to dispatcher which produces optimal schedule =

3, 2, 6, 11, 1, 7, 14, 4, 5, 9, 15, 10, 8, 19, 13, 18, 12, 22, 16, 17, 20, 23, 21, 24, 25

Results

Table A.15: Quality of Initial Solution, Problem 5

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average)	Standard Deviation in Rand. Initial Solution
SJF	47	43.79	3.054
LJF	45	42.38	2.668
MSLK	46	43.1	3.064
GRD	41	42.14	2.494
GRR	40	40.59	1.914
JER	41	43.59	3.519
LFT	39	42.06	2.745
EST	40	41.31	2.357
MCM	39	40.61	2.349
CCM	39	40.81	2.428
Average	41.7	42.038	2.694

The randomized initial solutions are based on the average of 100 single iteration randomized scheduling attempts. Figure A.13 shows a graphical comparison of the deterministic initial solution and the average randomized initial solution.

Figure A.13: Initial Solutions, Problem 5

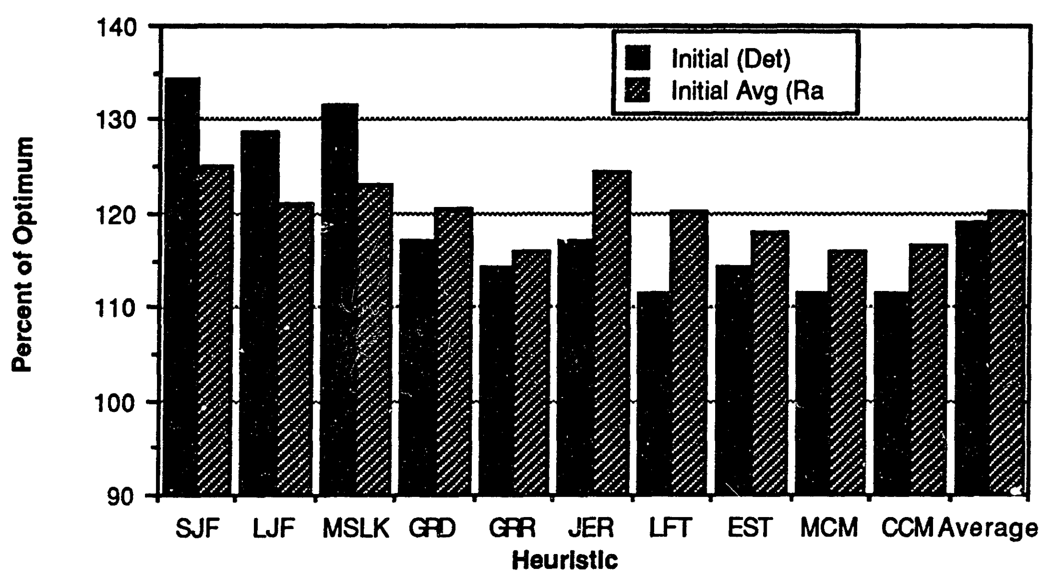


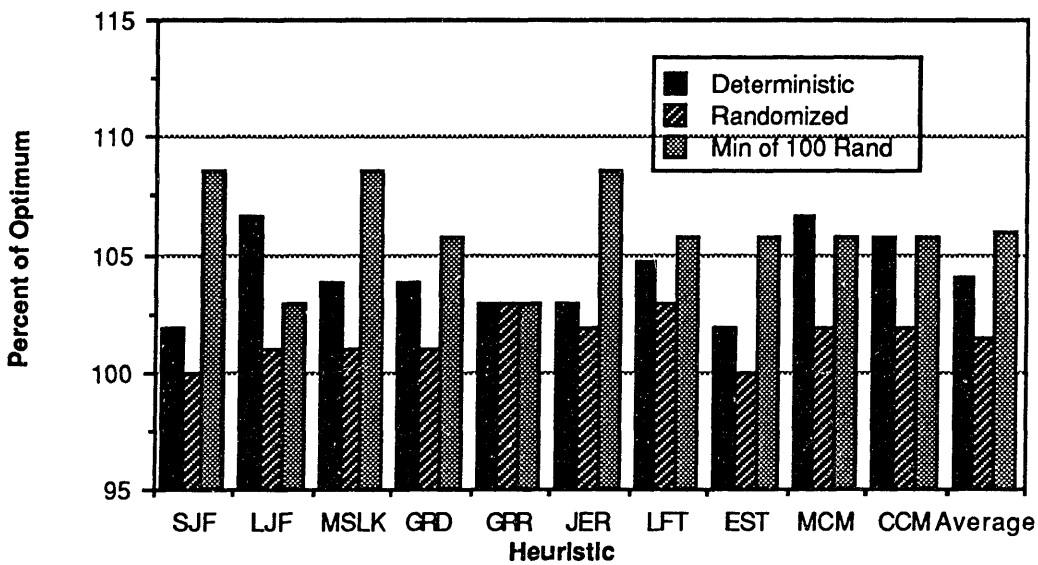
Table A.16: Average Solution after 100 Iterations, Problem 5 (Figure A.14)

Heuristic	Average After 100 Iterations Deterministic	Randomized	Minimum After 100 Randomized Trials of 1 Iteration
1 SJF	35.667	35.000	38
2 LJF	37.333	35.333	36
3 MSLK	36.333	35.333	38
4 GRD	36.333	35.333	37
5 GRR	36.000	36.000	36
6 JER	36.000	35.667	38
7 LFT	36.667	36.000	37
8 EST	35.667	35.000	37
9 MCM	37.333	35.667	37
10 CCM	37.000	35.667	37
Average	36.4333	35.5	37.1

Of the 30 deterministic scheduling attempts (10 heuristics x 3 attempts per heuristic), 3 had converged to optimality after 50 iterations, and 5 had converged to optimality after 100 iterations.

Of the 30 randomized scheduling attempts, 9 had converged to optimality after 50 iterations, and 16 had converged to optimality after 100 iterations.

Figure A.14: Final Average Solution, Problem 5



A.6 Test Problem 6

This problem is from [Davis, 1969, p. 201].

Table A.17: Problem Data for Problem 6

Job	Duration	Resource Usage (Resource 1)	Resource Usage (Resource 2)	Resource Usage Resource 3
1	3	3	5	2
2	2	5	4	3
3	6	5	2	2
4	1	4	1	4
5	3	5	5	4
6	2	3	5	2
7	1	2	4	4
8	2	3	2	2
9	2	3	2	4
10	1	3	3	2
11	7	4	1	4
12	4	1	4	4
13	4	2	2	2
14	2	5	5	4
15	1	1	5	4
16	7	4	5	4
17	5	3	2	3
18	1	5	3	3
19	8	2	4	6
20	3	1	6	2
Resource Limits		6	10	10

Note: The resource usages and limits are exactly the same as for the first 20 jobs in Problems 4 and 5.

The following graph shows the precedence relations between the jobs. Nodes S and T are dummy nodes denoting the start and end of the schedule.

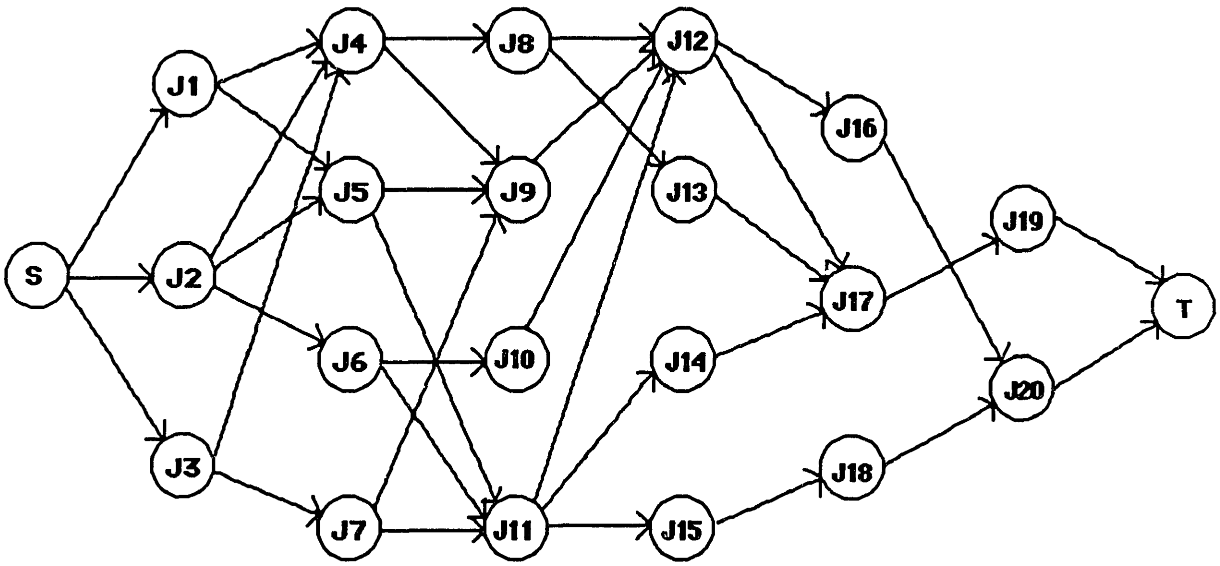


Figure A.15: Precedence Network for Problem 6

Problem 6: Summary Statistics

NUMBER OF PROJECTS	= 1
NUMBER OF NODES	= 22 INCLUDING 2 DUMMY NODES
NUMBER OF ARCS	= 35 INCLUDING 5 DUMMY ARCS
TOTAL ACTIVITY DENSITY	= 82
AVERAGE ACTIVITY DENSITY	= 3.73
COMPLEXITY	= 1.59

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 65
	Average	= 2.95
	Variance	= 4.64
	Maximum	= 8
CRITICAL PATH LENGTH	Sum	= 31
ACTIVITY SLACK	Total Slack	= 68
	# Of Activities With Positive Slack	= 14 (63.64%)
	Average Slack Per Activity	= 3.09
	Project Density - Total	= 0.49
	Total Slack Ratio	= 2.19
	Average Slack Ratio	= 0.10
FREE SLACK	Total	= 34
	# Of Activities With Positive Free Slack	= 7 (31.82%)
	Average Free Slack Per Activity	= 1.55
	Project Density - Free	= 0.66

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	NO
Audio/Vibration Resource?	NO
Number Of Other Resources	= 3

PERCENT OF DEMAND	Resource 1	= 0.91
	Resource 2	= 0.91
	Resource 3	= 0.91
	AVERAGE	= 0.91
	VARIANCE	= 0.0000
RESOURCE UTILIZATION	Resource 1	= 1.12
	Resource 2	= 0.71
	Resource 3	= 0.73
	AVERAGE	= 0.85
	VARIANCE	= 0.0546
AVG QTY WHEN DEMANDED	Resource 1	= 3.20
	Resource 2	= 3.50
	Resource 3	= 3.25
	AVERAGE	= 3.32
	VARIANCE	= 0.0258
RESOURCE CONSTRAINEDNESS	Resource 1	= 0.53
	Resource 2	= 0.35
	Resource 3	= 0.33
	AVERAGE	= 0.40
	VARIANCE	= 0.0129

	Resource 1 = 0.06
RES CONSTRAINEDNESS OVER TIME	Resource 2 = 0.04
	Resource 3 = 0.04
	AVERAGE = 0.04
	VARIANCE = 0.0001
RES CONSTR (ALL ACTIVITIES)	Resource 1 = 0.48
	Resource 2 = 0.32
	Resource 3 = 0.30
	AVERAGE = 0.37
	VARIANCE = 0.0107
OBSTRUCTION FACTOR	Resource 1 = 0.2679
	Resource 2 = 0.0545
	Resource 3 = 0.0133
	TOTAL = 0.3358
UNDERUTILIZATION FACTOR	Resource 1 = 0.1579
	Resource 2 = 0.4636
	Resource 3 = 0.3850
	TOTAL = 1.0065
EXCESS DEMAND TIME PERIODS	Resource 1 = 15.00
	Resource 2 = 6.00
	Resource 3 = 2.00
	AVERAGE = 7.67
	VARIANCE = 44.3333
TIME UNDERUTILIZATION	Resource 1 = 16.00
	Resource 2 = 25.00
	Resource 3 = 29.00
	AVERAGE = 23.33
	VARIANCE = 44.3333

Table A.18: Optimal Solution, Problem 6

The following solution is optimal, with a duration of 43.

Job	Start Time	End Time
1	2	5
2	0	2
3	8	14
4	14	15
5	5	8
6	2	4
7	14	15
8	15	17
9	15	17
10	4	5
11	17	24
12	24	28
13	17	21
14	24	26
15	26	27

Job	Start Time	End Time
16	33	40
17	28	33
18	27	28
19	33	41
20	40	43

Resource Level vs. Time for Optimal Solution:

Time	Res. 1	Res. 2	Res. 3
0	5	4	3
2	6	10	4
4	6	8	4
5	5	5	4
8	5	2	2
14	6	5	8
15	6	4	6
17	6	3	6
21	4	1	4
24	6	9	8
26	2	9	8
27	6	7	7
28	3	2	3
33	6	9	10
40	3	10	8
41	1	6	2
43	0	0	0

Job ordering to dispatcher which produces optimal schedule =

2, 1, 5, 3, 4, 7, 9, 6, 10, 8, 13, 11, 12, 14, 17, 19, 16, 15, 18, 20.

Results

Table A.19: Quality of Initial Solution, Problem 6

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average)	Standard Deviation in Rand. Initial Solution
1 SJF	43	47.930	3.308
2 LJF	53	51.300	3.176
3 MSLK	50	50.510	3.439
4 GRD	53	50.920	3.075
5 GRR	48	50.860	3.658
6 JER	50	49.810	3.258
7 LFT	46	50.480	3.300
8 EST	54	49.600	2.956
9 MCM	50	50.820	3.275
10 CCM	50	50.940	3.212
Average	49.7	50.314	3.271

The randomized initial solutions are based on the average of 100 single iteration randomized scheduling attempts. Figure A.16 shows a graphical comparison of the deterministic initial solution and the average randomized initial solution.

Figure A.16: Initial Solutions, Problem 6

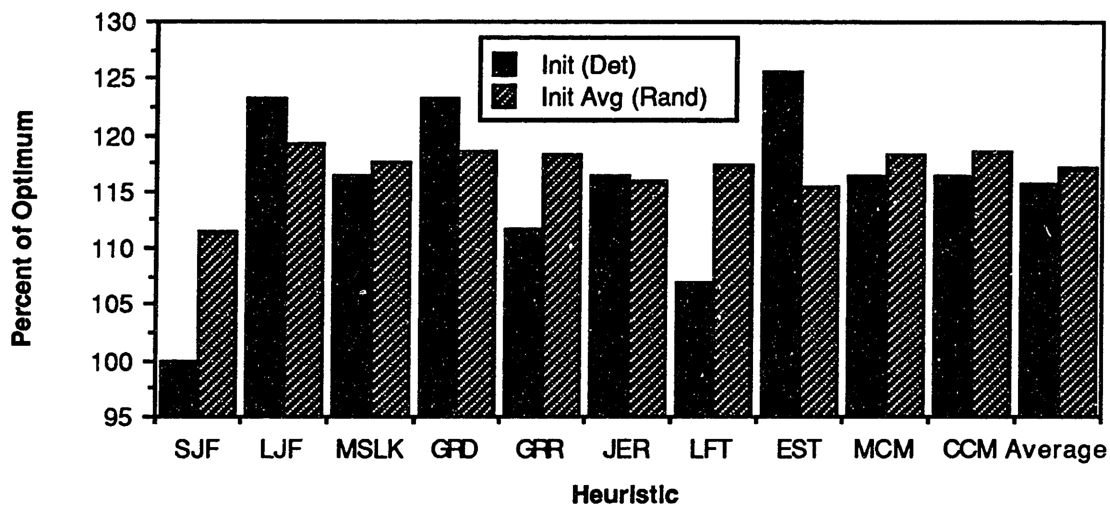


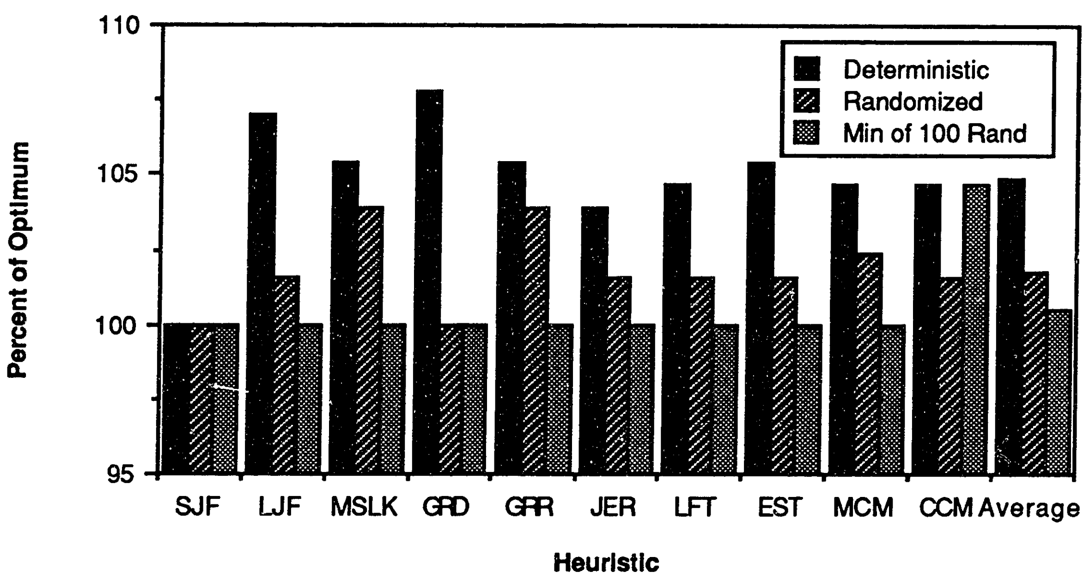
Table A.20: Average Solution after 100 Iterations, Problem 6 (Figure A.17)

Heuristic	Average After 100 Iterations: Deterministic	Average After 100 Iterations: Randomized	Minimum After 100 Randomized Trials of 1 Iteration
1 SJF	43.000	43.000	43
2 LJF	46.000	43.667	43
3 MSLK	45.333	44.667	43
4 GRD	46.333	43.000	43
5 GRR	45.333	44.667	43
6 JER	44.667	43.667	43
7 LFT	45.000	43.667	43
8 EST	45.333	43.667	43
9 MCM	45.000	44.000	43
10 CCM	45.000	43.667	45
Average	45.1	43.767	43.2

Of the 30 deterministic scheduling attempts (10 heuristics x 3 attempts per heuristic), 9 had converged to optimality after 50 iterations, and none of the other attempts had converged to optimality after 100 iterations.

Of the 30 randomized scheduling attempts, 19 had converged to optimality after 50 iterations, and none of the other attempts had converged to optimality after 100 iterations.

Figure A.17: Final Average Solution, Problem 6



A.7 Test Problem 7

This problem is from [Patterson and Davis, 1975]. Some of the nodes have been renumbered.

Table A.21: Problem Data for Problem 7

Job	Duration	Resource Usage (Resource 1)	Resource Usage (Resource 2)	Resource Usage Resource 3
1	5	3	5	2
2	5	5	4	3
3	3	5	2	2
4	4	4	1	4
5	2	2	4	4
6	1	5	5	4
7	6	3	5	2
8	6	3	2	2
9	3	4	1	4
10	3	3	2	4
11	3	3	3	2
12	3	1	4	4
13	3	5	5	4
14	6	2	2	2
15	4	1	5	4
16	3	5	3	3
17	3	3	2	3
18	4	4	5	4
19	1	2	4	6
20	4	1	0	4
21	4	1	6	2
22	1	2	2	2
23	6	3	2	1
24	3	2	2	2
25	3	0	1	3
Resource Limits		6	6	6

The following graph shows the precedence relations between the jobs. Nodes S and T are dummy nodes denoting the start and end of the schedule.

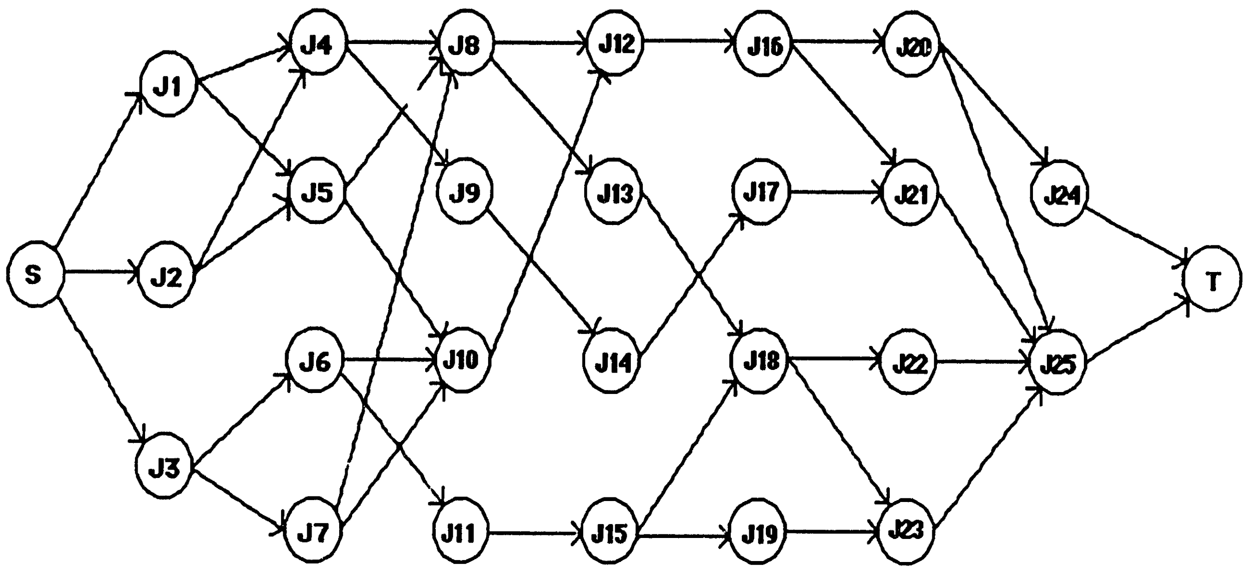


Figure A.18: Precedence Network for Problem 7

Problem 7: Summary Statistics

NUMBER OF PROJECTS	= 1
NUMBER OF NODES	= 27 INCLUDING 2 DUMMY NODES
NUMBER OF ARCS	= 40 INCLUDING 5 DUMMY ARCS
TOTAL ACTIVITY DENSITY	= 107
AVERAGE ACTIVITY DENSITY	= 3.96
COMPLEXITY	= 1.48

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 87
	Average	= 3.22
	Variance	= 2.38
	Maximum	= 6
CRITICAL PATH LENGTH	Sum	= 31
ACTIVITY SLACK	Total Slack	= 74
	# Of Activities With Positive Slack	= 15 (55.56%)
	Average Slack Per Activity	= 2.74
	Project Density - Total	= 0.54
	Total Slack Ratio	= 2.39
	Average Slack Ratio	= 0.09
FREE SLACK	Total	= 28
	# Of Activities With Positive Free Slack	= 7 (25.93%)
	Average Free Slack Per Activity	= 1.04
	Project Density - Free	= 0.76

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	NO
Audio/Vibration Resource?	NO
Number Of Other Resources	= 3

PERCENT OF DEMAND	Resource 1	= 0.89
	Resource 2	= 0.89
	Resource 3	= 0.93
	AVERAGE	= 0.90
	VARIANCE	= 0.0005
RESOURCE UTILIZATION	Resource 1	= 1.33
	Resource 2	= 1.46
	Resource 3	= 1.31
	AVERAGE	= 1.37
	VARIANCE	= 0.0064
AVG QTY WHEN DEMANDED	Resource 1	= 3.00
	Resource 2	= 3.21
	Resource 3	= 3.04
	AVERAGE	= 3.08
	VARIANCE	= 0.0122
RESOURCE CONSTRAINEDNESS	Resource 1	= 0.50
	Resource 2	= 0.53
	Resource 3	= 0.51
	AVERAGE	= 0.51
	VARIANCE	= 0.0003

RES CONSTRAINEDNESS OVER TIME	Resource 1	= 0.06
	Resource 2	= 0.06
	Resource 3	= 0.05
	AVERAGE	= 0.06
	VARIANCE	= 0.0000
RES CONSTR (ALL ACTIVITIES)	Resource 1	= 0.44
	Resource 2	= 0.48
	Resource 3	= 0.47
	AVERAGE	= 0.46
	VARIANCE	= 0.0003
OBSTRUCTION FACTOR	Resource 1	= 0.3548
	Resource 2	= 0.4133
	Resource 3	= 0.3333
	TOTAL	= 1.1015
UNDERUTILIZATION FACTOR	Resource 1	= 0.1048
	Resource 2	= 0.0996
	Resource 3	= 0.0988
	TOTAL	= 0.3032
EXCESS DEMAND TIME PERIODS	Resource 1	= 19.00
	Resource 2	= 22.00
	Resource 3	= 22.00
	AVERAGE	= 21.00
	VARIANCE	= 3.0000
TIME UNDERUTILIZATION	Resource 1	= 12.00
	Resource 2	= 9.00
	Resource 3	= 9.00
	AVERAGE	= 10.00
	VARIANCE	= 3.0000

Table A.22: Optimal Solution, Problem 7

The following solution is optimal, with a duration of 64.

Job	Start Time	End Time	Job	Start Time	End Time
1	0	5	16	54	57
2	5	10	17	51	54
3	16	19	18	40	44
4	10	14	19	36	37
5	14	16	20	57	61
6	19	20	21	57	61
7	20	26	22	44	45
8	26	32	23	48	54
9	44	45	24	61	64
10	29	32	25	61	64
11	26	29			
12	45	48			
13	37	40			
14	45	51			
15	32	36			

Resource Level vs. Time for Optimal Solution:

Time	Res. 1	Res. 2	Res. 3
0	3	5	2
5	5	4	3
10	4	1	4
14	2	4	4
16	5	2	2
19	5	5	4
20	3	5	2
26	6	5	4
29	6	4	6
32	1	5	4
36	2	4	6
37	5	5	4
40	4	5	4
44	6	3	5
45	3	6	6
48	5	4	3
51	6	4	4
54	5	3	3
57	2	6	6
61	2	3	5
64	0	0	0

Job ordering to dispatcher which produces optimal schedule =

1, 2, 4, 5, 3, 6, 7, 8, 11, 15, 10, 19, 13, 18, 9, 14, 17, 12, 16, 20, 24, 23, 21, 22, 25.

Results

Table A.23: Quality of Initial Solution, Problem 7

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average)	Standard Deviation in Rand. Initial Solution
1 SJF	74.000	72.46	2.94
2 LJF	80.000	73.81	3.70
3 MSLK	77.000	72.23	3.23
4 GRD	76.000	72.39	3.16
5 GRR	73.000	71.80	2.88
6 JER	73.000	72.357	3.13
7 LFT	70.000	72.04	2.77
8 EST	70.000	71.92	3.05
9 MCM	70.000	72.92	3.15
10 CCM	77.000	72.64	2.93
Average	74.000	72.457	

The randomized initial solutions are based on the average of 100 single iteration randomized scheduling attempts, except for Heuristic 6, which is based on 1000 single iteration randomized scheduling attempts. Figure A.19 shows a graphical comparison of the deterministic initial solution and the average randomized initial solution.

Figure A.19: Initial Solutions, Problem 7

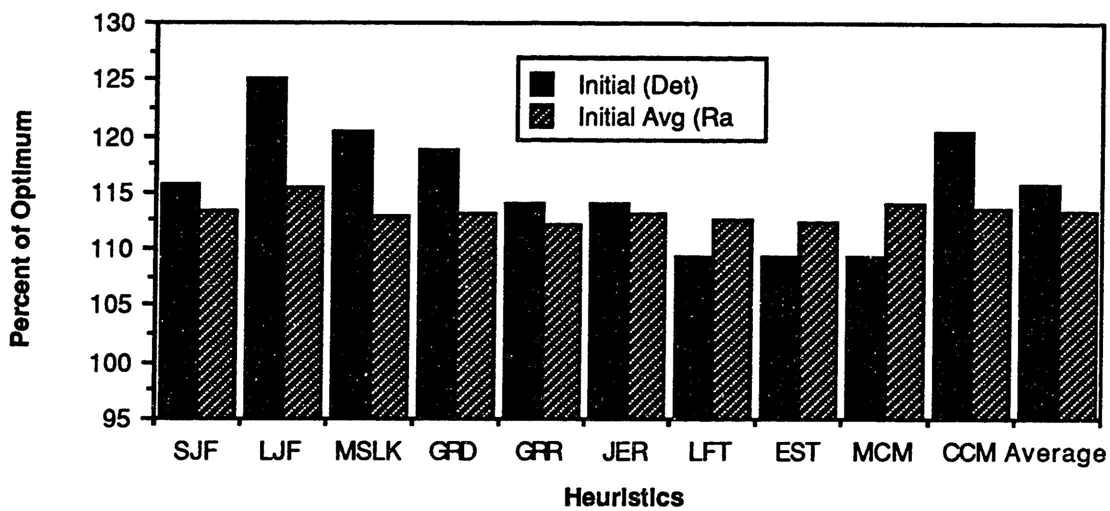
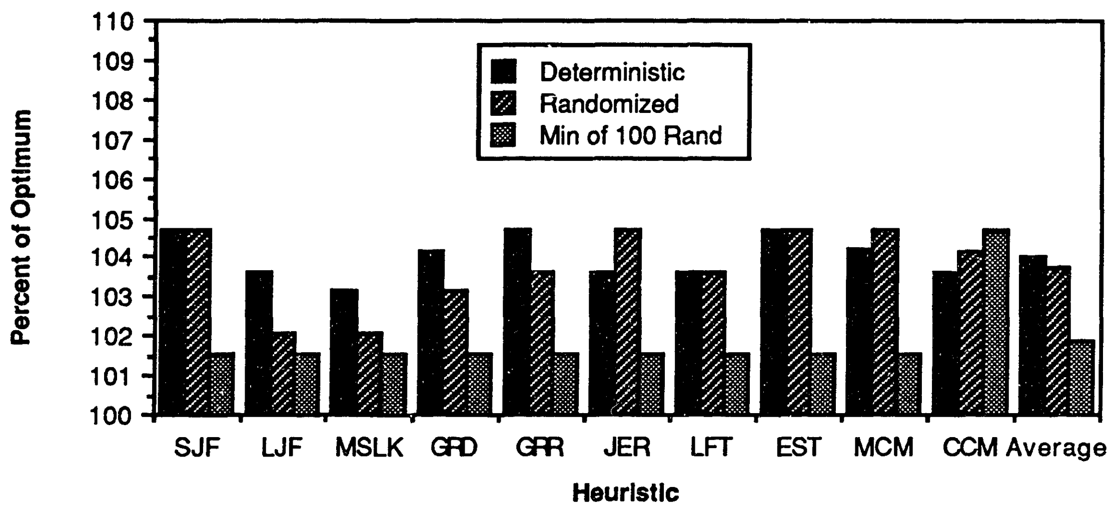


Table A.24: Average Solution after 100 Iterations, Problem 7 (Figure A.20)

Heuristic	Average After 100 Iterations Deterministic	Randomized	Minimum After 100 Randomized Trials of 1 Iteration
1 SJF	67	67	65
2 LJF	66.333333	65.333333	65
3 MSLK	66	65.333333	65
4 GRD	66.666667	66	65
5 GRR	67	66.333333	65
6 JER	66.333333	67	65*
7 LFT	66.333333	66.333333	65
8 EST	67	67	65
9 MCM	66.7	67	65
10 CCM	66.3	66.666667	67
Average	66.567	66.400	65.200

*this solution was found 15 times in 1000 iterations.

Figure A.20: Final Average Solution, Problem 7



Of the 67 deterministic scheduling attempts (20 attempts each for Heuristics 9 and 10, 6 attempts for Heuristic 6, and 3 attempts for each of the other heuristics), 6 had converged to optimality after 50 iterations, and none of the other attempts had converged to optimality after 100 iterations.

Of the 67 randomized scheduling attempts, 3 had converged to optimality after 50 iterations, and none of the other attempts had converged to optimality after 100 iterations.

Instead of taking 100 iterations of an algorithm, the same amount of computational work would be necessary to take 2 trial of 50 iterations each and then to take them minimum of the resulting values. This was calculated for each of the 10 heuristics applied to this problem. Without exception, the average results obtained after finding the minimum of two 50 iteration trials were equal to or better than the average solution after one 100 iteration trial. For the deterministic algorithms, the average was 66.162, and for the randomized algorithms, the average was 65.967.

A.8 Test Problem 8

This problem is from [Stinson, 1973].

Table A.25: Problem Data for Problem 8

Job	Duration	Resource Usage (Resource 1)	Resource Usage (Resource 2)	Resource Usage (Resource 3)	Resource Usage (Resource 4)
1	4	3	2	4	2
2	6	6	2	3	1
3	6	5	4	3	2
4	2	1	3	3	1
5	2	4	1	6	2
6	3	4	1	4	5
7	5	1	2	2	2
8	7	1	5	2	3
9	6	6	1	5	3
10	8	6	1	6	3
11	2	5	1	6	6
12	1	3	4	4	3
13	7	1	1	1	1
14	2	1	1	4	1
15	1	1	1	2	3
16	9	1	3	5	2
17	5	4	2	3	1
18	3	5	1	4	3
19	3	2	2	5	3
20	4	3	5	6	5
21	5	2	6	6	1
22	1	3	2	4	1
23	1	2	6	4	3
24	1	1	1	2	4
25	5	6	3	6	1
26	6	6	2	2	6
27	3	5	1	5	1
28	3	3	3	2	4
29	7	4	3	4	3
30	5	2	5	3	3
31	2	1	3	5	5
32	3	2	3	3	2
33	6	4	1	4	1
34	7	3	5	5	4
35	1	1	4	5	4
36	3	1	5	1	2
37	2	5	3	1	3
38	2	2	4	3	3
39	1	6	3	2	3

40	2	2	6	5	4
41	3	3	3	2	4
Resource Limits		10	10	10	10

The following graph shows the precedence relations between the jobs. Nodes S and T are dummy nodes denoting the start and end of the schedule.

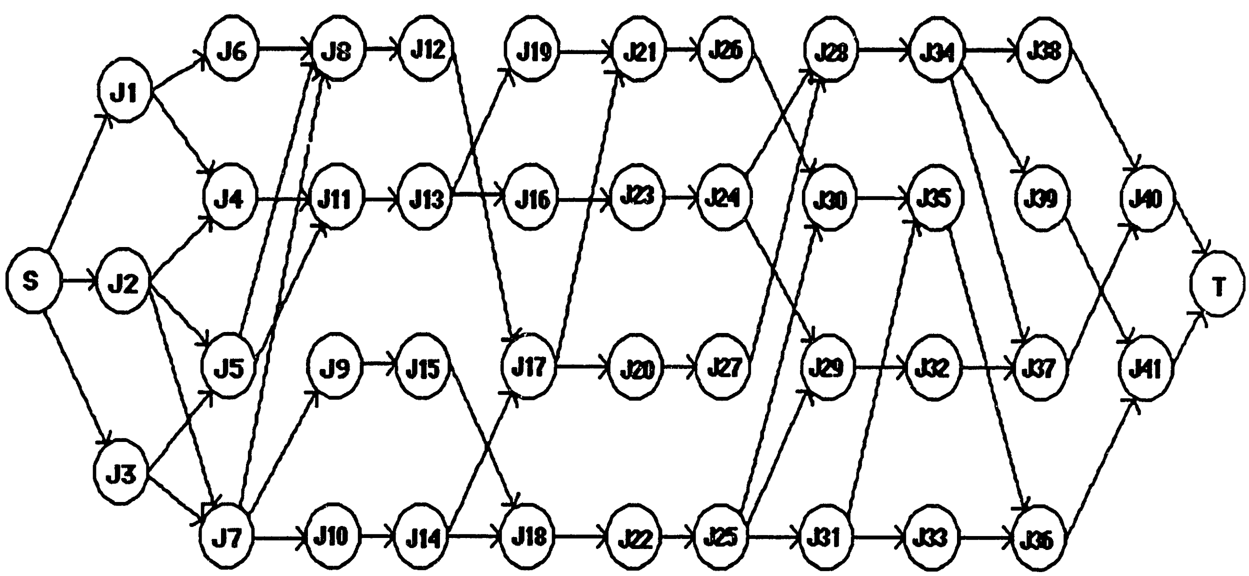


Figure A.21: Precedence Network for Problem 8

Problem 8: Summary Statistics

NUMBER OF PROJECTS	= 1
NUMBER OF NODES	= 43 INCLUDING 2 DUMMY NODES
NUMBER OF ARCS	= 60 INCLUDING 5 DUMMY ARCS
TOTAL ACTIVITY DENSITY	= 304
AVERAGE ACTIVITY DENSITY	= 7.07
COMPLEXITY	= 1.40

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 155
	Average	= 3.60
	Variance	= 4.86
	Maximum	= 9
CRITICAL PATH LENGTH	Sum	= 49
ACTIVITY SLACK	Total Slack	= 133
	# Of Activities With Positive Slack	= 29 (67.44%)
	Average Slack Per Activity	= 3.09
	Project Density - Total	= 0.54
	Total Slack Ratio	= 2.71
	Average Slack Ratio	= 0.06
FREE SLACK	Total	= 29
	# Of Activities With Positive Free Slack	= 9 (20.93%)
	Average Free Slack Per Activity	= 0.67
	Project Density - Free	= 0.84

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	NO
Audio/Vibration Resource?	NO
Number Of Other Resources	= 4

PERCENT OF DEMAND	Resource 1	= 0.95
	Resource 2	= 0.95
	Resource 3	= 0.95
	Resource 4	= 0.95
	AVERAGE	= 0.95
	VARIANCE	= 0.0000
RESOURCE UTILIZATION	Resource 1	= 1.05
	Resource 2	= 0.88
	Resource 3	= 1.20
	Resource 4	= 0.83
	AVERAGE	= 0.99
	VARIANCE	= 0.0274
AVG QTY WHEN DEMANDED	Resource 1	= 3.10
	Resource 2	= 2.80
	Resource 3	= 3.71
	Resource 4	= 2.78
	AVERAGE	= 3.10
	VARIANCE	= 0.1860
RESOURCE CONSTRAINEDNESS	Resource 1	= 0.31
	Resource 2	= 0.28
	Resource 3	= 0.37
	Resource 4	= 0.28
	AVERAGE	= 0.31
	VARIANCE	= 0.0019

RES CONSTRAINEDNESS OVER TIME	Resource 1	= 0.03
	Resource 2	= 0.02
	Resource 3	= 0.03
	Resource 4	= 0.02
	AVERAGE	= 0.02
	VARIANCE	= 0.0000
RES CONSTR (ALL ACTIVITIES)	Resource 1	= 0.30
	Resource 2	= 0.27
	Resource 3	= 0.35
	Resource 4	= 0.27
	AVERAGE	= 0.30
	VARIANCE	= 0.0017
OBSTRUCTION FACTOR	Resource 1	= 0.2155
	Resource 2	= 0.1085
	Resource 3	= 0.2440
	Resource 4	= 0.0711
	TOTAL	= 0.6392
UNDERUTILIZATION FACTOR	Resource 1	= 0.1670
	Resource 2	= 0.2402
	Resource 3	= 0.0802
	Resource 4	= 0.2721
	TOTAL	= 0.7594
EXCESS DEMAND TIME PERIODS	Resource 1	= 28.00
	Resource 2	= 13.00
	Resource 3	= 30.00
	Resource 4	= 10.00
	AVERAGE	= 20.25
	VARIANCE	= 104.2500
TIME UNDERUTILIZATION	Resource 1	= 21.00
	Resource 2	= 36.00
	Resource 3	= 19.00
	Resource 4	= 39.00
	AVERAGE	= 28.75
	VARIANCE	= 104.2500

Table A.26: Optimal Solution, Problem 8

The following solution is optimal, with a duration of 74.

Job	Start Time	End Time	Job	Start Time	End Time
1	0	4	22	39	40
2	0	6	23	43	44
3	6	12	24	44	45
4	6	8	25	43	48
5	12	14	26	55	61
6	4	7	27	40	43
7	12	17	28	45	48
8	17	24	29	48	55
9	25	31	30	61	66
10	17	25	31	53	55
11	14	16	32	62	65
12	24	25	33	62	68
13	16	23	34	55	62
14	25	27	35	67	68
15	31	32	36	68	71
16	27	36	37	67	69
17	31	36	38	65	67
18	36	39	39	69	70
19	40	43	40	71	73
20	36	40	41	71	74
21	48	53			

Resource Level vs. Time for Optimal Solution:

Time	Res. 1	Res. 2	Res. 3	Res. 4
0	9	4	7	3
4	10	3	7	6
6	10	8	10	8
7	6	7	6	3
8	5	4	3	2
12	5	3	8	4
14	6	3	8	8
16	2	3	3	3
17	8	7	9	7
23	7	6	8	6
24	9	5	10	6
25	7	2	9	4
27	7	4	10	5
31	6	6	10	6
32	5	5	8	3
36	8	6	10	8
39	6	7	10	6
40	7	3	10	4
43	8	9	10	4
44	7	4	8	5
45	9	6	8	5
48	6	9	10	4
53	5	6	9	8
55	9	7	7	10
61	5	10	8	7
62	8	9	10	6
65	8	10	10	7
66	6	5	7	4
67	10	8	10	8
68	6	8	2	5
69	7	8	3	5
70	1	5	1	2
71	5	9	7	8
73	3	3	2	4
74	0	0	0	0

Job ordering to dispatcher which produces optimal schedule =

2, 1, 6, 3, 4, 5, 7, 11, 13, 8, 10, 12, 9, 14, 16, 15, 17, 18, 20, 22, 19, 27, 23, 25, 24, 28, 21, 29, 31, 26, 34, 30, 32, 33, 38, 35, 37, 36, 39, 40, 41.

Results

Table A.27: Quality of Initial Solution, Problem 8

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average)	Standard Deviation in Rand. Initial Solution
5 GRR	78	86.46	4.681
9 MCM	81	85.40	3.758

The randomized initial solutions are based on the average of 100 single iteration randomized scheduling attempts. Only two of the heuristics were tested on this problem.

Table A.28: Average Solution after 100 Iterations, Problem 8

	Average After 100 Iterations		Minimum After 100 Randomized
Heuristic	Deterministic	Randomized	Trials of 1 Iteration
5 GRR	77.667	77.667	77
9 MCM	77	77.333	78

A.9 Test Problem 9

This problem was created to demonstrate scheduling where crewmembers have different processing times, where jobs may require multiple crewmembers, and to utilize the audio/vibration nonlinear resource constraint. There are no time constraints (e.g. no precedence constraints) on any of the jobs.

Table A.29: Problem Data for Problem 9

JOB NAMES	CREW REQ.	MIN. TIME	RES. 1	RES. 2	RES. 3	RES. 4	AUDIO LEVEL
1 Solar Astronomy	2	196	922	2601	2980	1340	GENERATING
2 Materials Processing	5	85	6	3991	2542	493	GENERATING
3 Exercise	3	19	378	107	1732	1473	GENERATING
4 EVA	3	69	89	694	668	677	NEUTRAL
5 Deploy Satellite	3	120	789	1794	134	498	SENSITIVE
6 Medical Diagnostics	1	19	171	466	56	998	SENSITIVE
7 Eat Dinner	4	40	317	3050	1179	1081	GENERATING
8 UV Astronomy	4	44	659	361	1035	47	SENSITIVE
9 Clean Space Station	1	102	795	3037	350	303	NEUTRAL
10 Sleep Period 1	1	161	102	2957	1636	364	SENSITIVE
11 Sleep Period 2	2	70	998	2113	2917	1468	NEUTRAL
12 Atmospheric Observation	2	100	505	411	2044	378	GENERATING
13 Plan Daily Timelines	4	228	77	285	1971	1021	SENSITIVE
14 Repair Satellite	1	108	480	1209	1354	1451	GENERATING
15 Plant Growth Experiment	2	19	279	660	2520	658	NEUTRAL
16 Earth Observation Experiment	4	165	513	2481	1684	1660	NEUTRAL
17 IR Astronomy	2	135	24	3637	260	1908	NEUTRAL
18 Eat Breakfast	4	30	532	965	2898	1252	GENERATING
19 Fluids Management Experiment	1	189	285	911	598	784	GENERATING

The limits on the 4 resources in this problem are 1500, 6000, 4500, and 3000 respectively.

For a job which requires n crewmembers, the Min. Time is computed by finding the n<u>th</u> smallest performance time, among those crewmembers rated for the job.

CREWMEMBER PERFORMANCE TIMES (MINUTES)

	JOBS	CREW 1	CREW 2	CREW 3	CREW 4	CREW 5
1	Solar Astronomy	200	190	*	230	196
2	Materials Processing	75	65	80	85	70
3	Exercise	25	19	19	19	19
4	EVA	45	55	69	69	75
5	Deploy Satellite	95	85	120	*	120
6	Medical Diagnostics	*	30	29	30	19
7	Eat Dinner	40	40	40	40	40
8	UV Astronomy	40	44	40	44	44
9	Clean Space Station	102	102	102	102	102
10	Sleep Period 1	161	161	161	161	161
11	Sleep Period 2	70	70	70	70	70
12	Atmospheric Observation	90	100	100	100	100
13	Plan Daily Timelines	228	228	200	228	228
14	Repair Satellite	108	120	108	125	108
15	Plant Growth Experiment	35	19	18	19	19
16	Earth Observation Experiment	165	165	165	165	165
17	IR Astronomy	135	145	130	144	144
18	Eat Breakfast	40	30	24	24	24
19	Fluids Management Experiment	189	216	216	216	216

An * indicates that this crewmember is not rated to perform this job.

NUMBER OF PROJECTS	= 19
NUMBER OF NODES	= 19
NUMBER OF ARCS	= 0
NUMBER OF DUMMY ACTIVITIES	= 0
TOTAL ACTIVITY DENSITY	= 0
AVERAGE ACTIVITY DENSITY	= 0.00
COMPLEXITY	= 0.00

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 1899
	Average	= 99.95
	Variance	= 4253.61
	Maximum	= 228
CRITICAL PATH LENGTH	Sum	= 228
	Average	= 12.00
	Variance	= 2592.00
	Maximum	= 228
ACTIVITY SLACK	Total Slack	= 2433
	# Of Activities With Positive Slack	= 18 (94.74%)
	Average Slack Per Activity	= 128.05
	Project Density - Total	= 0.44
	Total Slack Ratio	= 10.67
	Average Slack Ratio	= 0.56
FREE SLACK	Total	= 2433
	# Of Activities With Positive Free Slack	= 18 (94.74%)
	Average Free Slack Per Activity	= 128.05
	Project Density - Free	= 0.44

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	YES
Audio/Vibration Resource?	YES
Number Of Other Resources	= 4

PERCENT OF DEMAND	Crew	= 1.00
	Resource 1	= 1.00
	Resource 2	= 1.00
	Resource 3	= 1.00
	Resource 4	= 1.00
	AVERAGE	= 1.00
	VARIANCE	= 0.0000
RESOURCE UTILIZATION	Crew	= 4.12
	Resource 1	= 2.29
	Resource 2	= 2.59
	Resource 3	= 2.78
	Resource 4	= 2.71
	AVERAGE	= 2.90
	VARIANCE	= 0.5001
AVG QTY WHEN DEMANDED	Crew	= 2.58
	Resource 1	= 416.89
	Resource 2	= 1670.00
	Resource 3	= 1503.05
	Resource 4	= 939.68
	AVERAGE	= 906.44
	VARIANCE	= 499173.7626

RESOURCE CONSTRAINEDNESS	Crew	0.52
	Resource 1	0.28
	Resource 2	0.28
	Resource 3	0.33
	Resource 4	0.31
	AVERAGE	0.34
	VARIANCE	0.0098
RES CONSTRAINEDNESS OVER TIME	Crew	0.22
	Resource 1	0.12
	Resource 2	0.14
	Resource 3	0.15
	Resource 4	0.14
	AVERAGE	0.15
	VARIANCE	0.0014
RES CONSTR (ALL ACTIVITIES)	Crew	0.52
	Resource 1	0.28
	Resource 2	0.28
	Resource 3	0.33
	Resource 4	0.31
	AVERAGE	0.34
	VARIANCE	0.0098
OBSTRUCTION FACTOR	Crew	0.7641
	Resource 1	0.6331
	Resource 2	0.6868
	Resource 3	0.6688
	Resource 4	0.6674
	TOTAL	3.4201
UNDERUTILIZATION FACTOR	Crew	0.0068
	Resource 1	0.0691
	Resource 2	0.0726
	Resource 3	0.0284
	Resource 4	0.0366
	TOTAL	0.2136
EXCESS DEMAND TIME PERIODS	Crew	196.00
	Resource 1	165.00
	Resource 2	165.00
	Resource 3	196.00
	Resource 4	189.00
	AVERAGE	182.20
	VARIANCE	254.7000
TIME UNDERUTILIZATION	Crew	32.00
	Resource 1	63.00
	Resource 2	63.00
	Resource 3	32.00
	Resource 4	39.00
	AVERAGE	45.80
	VARIANCE	254.7000

Table A.30: Optimal Solution, Problem 9

The following solution is believed (but not guaranteed) to be optimal, with a duration of 1047.

The table below shows a time line in the rightmost column. The first five columns show what job each of the crewmembers are doing at the corresponding time in the timeline, until the next time listed in the timeline (a 0 corresponds to no job at that time). The next four columns in the table indicate resource usage at the times given, and the tenth column indicates a -1 if a noise/vibration sensitive activity is occurring, a 1 if a noise/vibration generating activity is occurring, and a 0 otherwise. For example, it can be seen that from time 723 to time 742 Crewmembers 1 and 5 are performing Job 12, and Crewmembers 2, 3, and 4 are performing Job 3. Also at this time, the resource levels are 883, 518, 3776, and 1851. The 1 in the tenth column indicates that at least one of the ongoing jobs is noise generating. Following this timeline is a graphical representation of the schedule, Figure A.22.

CREW1	CREW2	CREW3	CREW4	CREW5	RES1	RES2	RES3	RES4	AUDIO	TIME
10	13	13	13	13	179	3242	3607	1385	-1	0
9	13	13	13	13	872	3322	2321	1324	-1	161
9	13	0	13	13	872	3322	2321	1324	-1	200
9	16	16	16	16	1308	5518	2034	1963	0	228
19	16	16	16	16	798	3392	2282	2444	1	263
19	1	0	0	1	1207	3512	3578	2124	1	393
0	1	14	0	1	1402	3810	4334	2791	1	452
4	1	4	4	1	1011	3295	3648	2017	1	560
4	0	4	4	1	1011	3295	3648	2017	1	583
4	11	4	4	11	1087	2807	3585	2145	0	589
0	11	4	4	11	1087	2807	3585	2145	0	605
0	11	6	0	11	1169	2579	2973	2466	-1	629
0	11	0	0	11	998	2113	2917	1468	0	658
0	18	18	18	18	532	965	2898	1252	1	659
7	18	7	7	7	849	4015	4077	2333	1	683
7	0	7	7	7	317	3050	1179	1081	1	689
12	3	3	3	12	883	518	3776	1851	1	723
12	0	17	17	12	529	4048	2304	2286	1	742
0	0	17	17	12	529	4048	2304	2286	1	813
5	5	17	17	5	813	5431	394	2406	-1	823
5	5	0	17	5	813	5431	394	2406	-1	872
5	5	15	15	5	1068	2454	2654	1156	-1	886
5	5	0	15	5	1068	2454	2654	1156	-1	904
5	5	0	0	5	789	1794	134	498	-1	905
5	0	0	0	5	789	1794	134	498	-1	908
8	8	8	8	5	1448	2155	1169	545	-1	918
8	8	8	8	0	659	361	1035	47	-1	943
0	8	0	8	0	659	361	1035	47	-1	958
2	2	2	2	2	6	3991	2542	493	1	962
2	0	2	2	2	6	3991	2542	493	1	1027
2	0	2	2	0	6	3991	2542	493	1	1032
0	0	2	2	0	6	3991	2542	493	1	1037
0	0	0	2	0	6	3991	2542	493	1	1042
0	0	0	0	0	0	0	0	0	0	1047

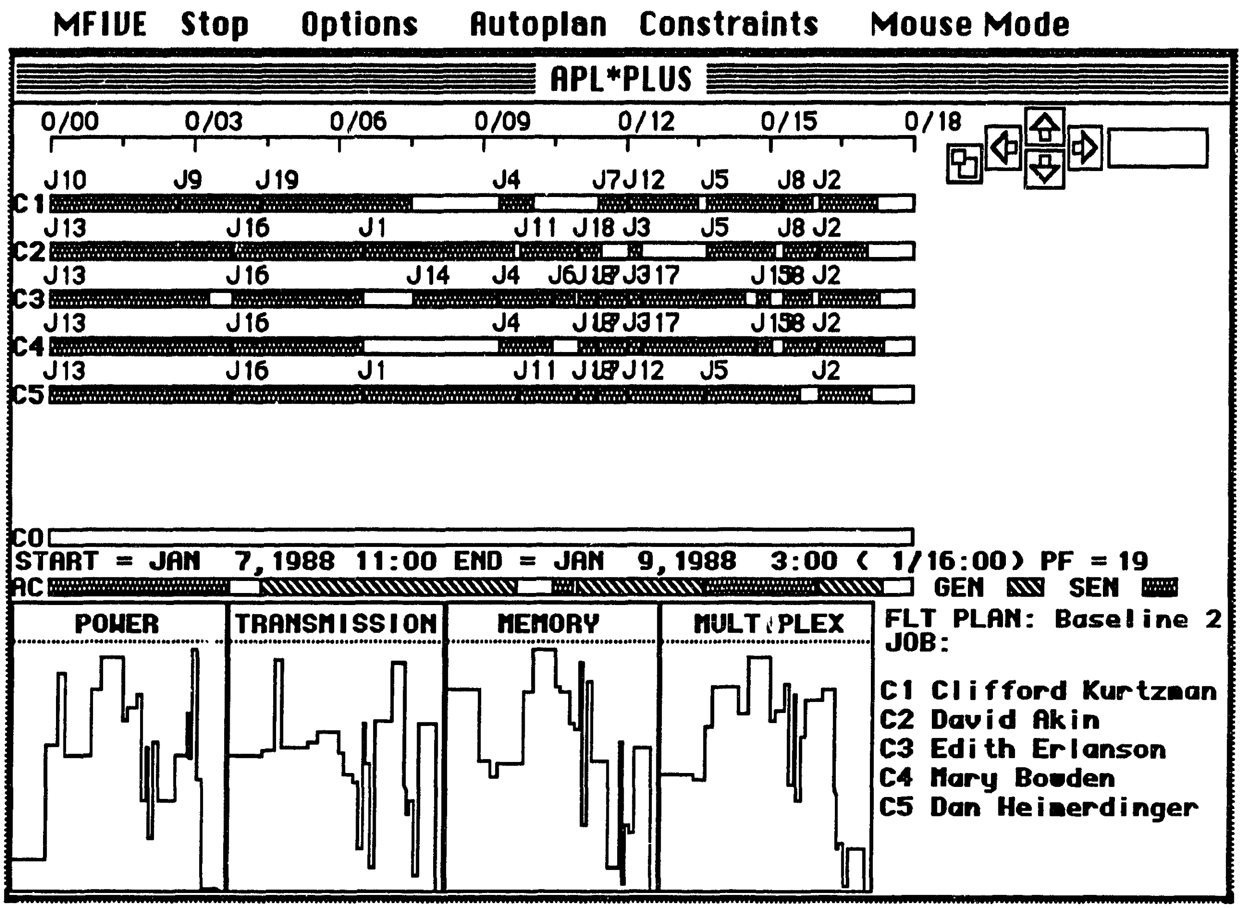


Figure A.22: Graphical Timeline For Problem 9

Job ordering to dispatcher which produces optimal schedule =
10, 13, 9, 16, 19, 1, 14, 4, 11, 18, 7, 3, 12, 17, 5, 8, 2, 15, 6.

Results

Table A.31: Quality of Initial Solution, Problem 9

Heuristic	Deterministic Initial Solution	Randomized Initial Solution (Average)	Standard Deviation in Rand. Initial Solution
1 SJF	1242	1323.97	
2 LJF	1196	1251.06	
3 MSLK	1196	1285.94	
4 GRD	1201	1242.48	
5 GRR	1201	1243.71	
6 JER	1547	1297.368	
7 LFT	1547	1294.53	
8 EST	1547	1293.54	80.241
9 MCM	1219	1197.08	51.098
10 CCM	1122	1209.51	56.107
Average	1301.800	1263.919	

(Standard Deviations were not obtained for the first 7 heuristics.)

The randomized initial solutions are based on the average of 100 single iteration randomized scheduling attempts, except for Heuristic 6, which is based on 1000 single iteration randomized scheduling attempts. Figure A.23 shows a graphical comparison of the deterministic initial solution and the average randomized initial solution.

It should be noted that as applied this problem, Heuristics 4 and 5 are identical, as are Heuristics 6 and 7. It is therefore not surprising that the results from each of the heuristics in each of the pairs are so similar to each other.

Figure A.23: Initial Solutions, Problem 9

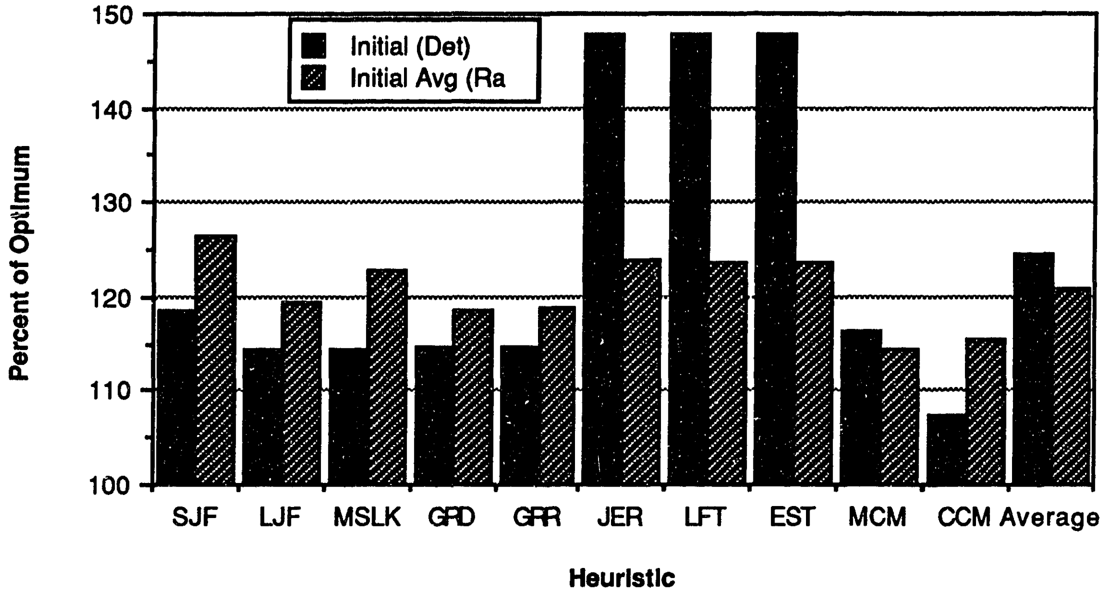


Table A.32: Average Solution after 100 Iterations, Problem 9 (Figure A.24)

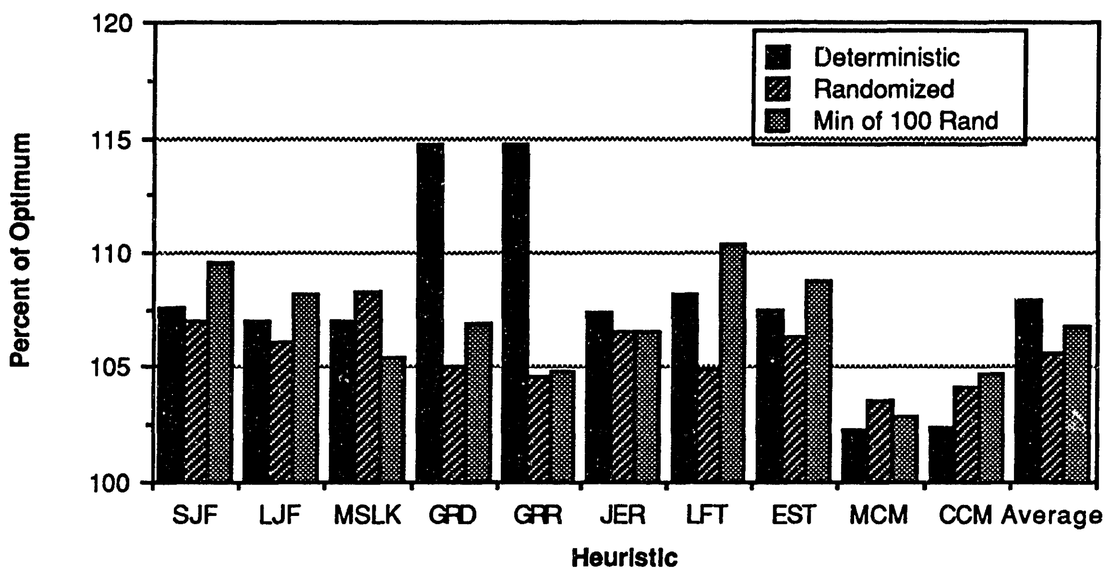
Heuristic	Average After 100 Iterations Deterministic	Randomized	Minimum After 100 Randomized Trials of 1 Iteration
1 SJF	1126.666667	1121	1147
2 LJF	1120	1111.333333	1132
3 MSLK	1121	1133.333333	1103
4 GRD	1201	1100	1119
5 GRR	1201	1095.333333	1097
6 JER	1124.5	1115.333333	1115*
7 LFT	1132.666667	1098.333333	1155
8 EST	1125	1112.666667	1139
9 MCM	1070.95	1084.333333	1077
10 CCM	1072.65	1090.333333	1096
Average	1129.543	1106.200	1118

*Based on 1000 Trials of 1 iteration

The averages in the first 2 columns are based on 20 attempts of 100 iterations each for Heuristics 9 and 10, 6 attempts for Heuristic 6, and 3 attempts for each of the other heuristics.

Instead of taking 100 iterations of an algorithm, the same amount of computational work would be necessary to take 2 trial of 50 iterations each and then to take them minimum of the resulting values. This was calculated for each of the 10 heuristics applied to this problem. For the deterministic algorithms, the average was 1130.603, and for the randomized algorithms, the average was 1109.467. Both results are slightly worse than the above results for 100 iteration trials.

Figure A.24: Final Average Solution, Problem 9



A.10 Test Problem 10

This problem was created to demonstrate scheduling where crewmembers have different processing times, where jobs may require multiple crewmembers, to utilize the audio/vibration nonlinear resource constraint, and to utilize complex time and target constraints.

Table A.33: Problem Data for Problem 10

This problem is in every respect identical to Problem 9, except that it has several additional time constraints, as indicated below:

Precedence Constraints:

Job 6 must precede Job 8.

Job 7 must precede Job 6.

Job 11 must precede Job 10.

Job 12 must precede Job 10.

Concurrence Constraints:

Job 14 and Job 19 must start concurrently.

Earliest Start Time of Job 2 is at 120 minutes after the beginning of the schedule.

Latest Start Time of Job 8 is 480 minutes after the beginning of the schedule.

Latest End Time of Job 4 is 420 minutes after the beginning of the schedule.

Maximum Difference in Start Time Constraints:

Job 15 must start no more than 480 minutes after Job 16.

Job 16 must start no more than 480 minutes after Job 15.

Job 17 must start no more than 480 minutes after Job 16.

Target Constraints:

No part of Job 7 can be scheduled during the interval from 60 to 180 minutes after the beginning of the schedule, nor during the interval from 540 to 660 minutes after the beginning of the schedule.

No part of Job 13 can be scheduled during the interval from 240 to 300 minutes after the beginning of the schedule.

NUMBER OF PROJECTS	= 12
NUMBER OF NODES	= 19
NUMBER OF ARCS	= 11
NUMBER OF DUMMY ACTIVITIES	= 0
TOTAL ACTIVITY DENSITY	= 4
AVERAGE ACTIVITY DENSITY	= 0.21
COMPLEXITY	= 0.58

STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS

ACTIVITY DURATION	Sum	= 1899
	Average	= 99.95
	Variance	= 4253.61
	Maximum	= 228
CRITICAL PATH LENGTH	Sum	= 261
	Average	= 21.75
	Variance	= 5203.69
	Maximum	= 261
ACTIVITY SLACK	Total Slack	= 2231
	# Of Activities With Positive Slack	= 17 (89.47%)
	Average Slack Per Activity	= 117.42
	Project Density - Total	= 0.46
	Total Slack Ratio	= 8.55
	Average Slack Ratio	= 0.45
FREE SLACK	Total	= 1843
	# Of Activities With Positive Free Slack	= 14 (73.68%)
	Average Free Slack Per Activity	= 97.00
	Project Density - Free	= 0.51

STATISTICS RELATING TO THE RESOURCES

Crew Resource?	YES
Audio/Vibration Resource?	YES
Number Of Other Resources	= 4

PERCENT OF DEMAND	Crew	= 1.00
	Resource 1	= 1.00
	Resource 2	= 1.00
	Resource 3	= 1.00
	Resource 4	= 1.00
	AVERAGE	= 1.00
	VARIANCE	= 0.0000
RESOURCE UTILIZATION	Crew	= 3.60
	Resource 1	= 2.00
	Resource 2	= 2.26
	Resource 3	= 2.43
	Resource 4	= 2.37
	AVERAGE	= 2.53
	VARIANCE	= 0.3816
AVG QTY WHEN DEMANDED	Crew	= 2.58
	Resource 1	= 416.89
	Resource 2	= 1670.00
	Resource 3	= 1503.05
	Resource 4	= 939.68
	AVERAGE	= 906.44
	VARIANCE	= 499173.7626

RESOURCE CONSTRAINEDNESS	Crew	= 0.52
	Resource 1	= 0.28
	Resource 2	= 0.28
	Resource 3	= 0.33
	Resource 4	= 0.31
	AVERAGE	= 0.34
	VARIANCE	= 0.0098
RES CONSTRAINEDNESS OVER TIME	Crew	= 0.19
	Resource 1	= 0.11
	Resource 2	= 0.12
	Resource 3	= 0.13
	Resource 4	= 0.12
	AVERAGE	= 0.13
	VARIANCE	= 0.0011
RES CONSTR (ALL ACTIVITIES)	Crew	= 0.52
	Resource 1	= 0.28
	Resource 2	= 0.28
	Resource 3	= 0.33
	Resource 4	= 0.31
	AVERAGE	= 0.34
	VARIANCE	= 0.0098
OBSTRUCTION FACTOR	Crew	= 0.7502
	Resource 1	= 0.6204
	Resource 2	= 0.6045
	Resource 3	= 0.6287
	Resource 4	= 0.6498
	TOTAL	= 3.2536
UNDERUTILIZATION FACTOR	Crew	= 0.0281
	Resource 1	= 0.1194
	Resource 2	= 0.0462
	Resource 3	= 0.0403
	Resource 4	= 0.0725
	TOTAL	= 0.3066
EXCESS DEMAND TIME PERIODS	Crew	= 205.00
	Resource 1	= 165.00
	Resource 2	= 205.00
	Resource 3	= 205.00
	Resource 4	= 196.00
	AVERAGE	= 195.20
	VARIANCE	= 300.2000
TIME UNDERUTILIZATION	Crew	= 56.00
	Resource 1	= 96.00
	Resource 2	= 56.00
	Resource 3	= 56.00
	Resource 4	= 65.00
	AVERAGE	= 65.80
	VARIANCE	= 300.2000

Table A.34: Optimal Solution, Problem 10

The following solution is believed (but not guaranteed) to be optimal, with a duration of 1104. The table below is interpreted similarly to the one presented for Problem 9. Following this timeline is a graphical representation of this schedule, Figure A.25.

CREW1	CREW2	CREW3	CREW4	CREW5	RES1	RES2	RES3	RES4	AUDIO	TIME
19	0	14	12	12	1270	2531	3996	2613	1	0
19	0	14	0	0	765	2120	1952	2235	1	100
19	1	0	0	1	1207	3512	3578	2124	1	108
4	1	4	4	1	1011	3295	3648	2017	1	189
0	1	4	4	1	1011	3295	3648	2017	1	234
0	1	0	0	1	922	2601	2980	1340	1	258
7	7	7	7	1	1239	5651	4159	2421	1	298
7	7	7	7	0	317	3050	1179	1081	1	304
0	0	11	11	6	1169	2579	2973	2466	¯1	338
0	0	11	11	0	998	2113	2917	1468	0	357
5	5	17	17	5	813	5431	394	2406	¯1	408
5	0	17	17	5	813	5431	394	2406	¯1	493
0	0	17	17	5	813	5431	394	2406	¯1	503
0	0	17	17	0	24	3637	260	1908	0	528
13	13	13	17	13	101	3922	2231	2929	¯1	538
13	13	13	10	13	179	3242	3607	1385	¯1	552
13	13	13	0	13	77	285	1971	1021	¯1	713
13	13	15	15	13	356	945	4491	1679	¯1	738
13	13	0	15	13	356	945	4491	1679	¯1	756
13	13	0	0	13	77	285	1971	1021	¯1	757
0	18	18	18	18	532	965	2898	1252	1	766
0	18	0	0	9	1327	4002	3248	1555	1	790
16	16	16	16	9	1308	5518	2034	1963	0	796
16	16	16	16	0	513	2481	1684	1660	0	892
8	0	8	8	8	659	361	1035	47	¯1	961
0	0	0	8	8	659	361	1035	47	¯1	1001
2	2	2	2	2	6	3991	2542	493	1	1005
2	0	2	2	2	6	3991	2542	493	1	1070
2	0	2	2	0	6	3991	2542	493	1	1075
0	0	2	2	0	6	3991	2542	493	1	1080
0	3	3	2	3	384	4098	4274	1966	1	1085
0	3	3	0	3	378	107	1732	1473	1	1090
0	0	0	0	0	0	0	0	0	0	1104

Job ordering to dispatcher which produces optimal schedule =
19, 14, 1, 7, 6, 11, 5, 17, 13, 18, 16, 8, 2, 3, 12, 10, 9, 4, 15

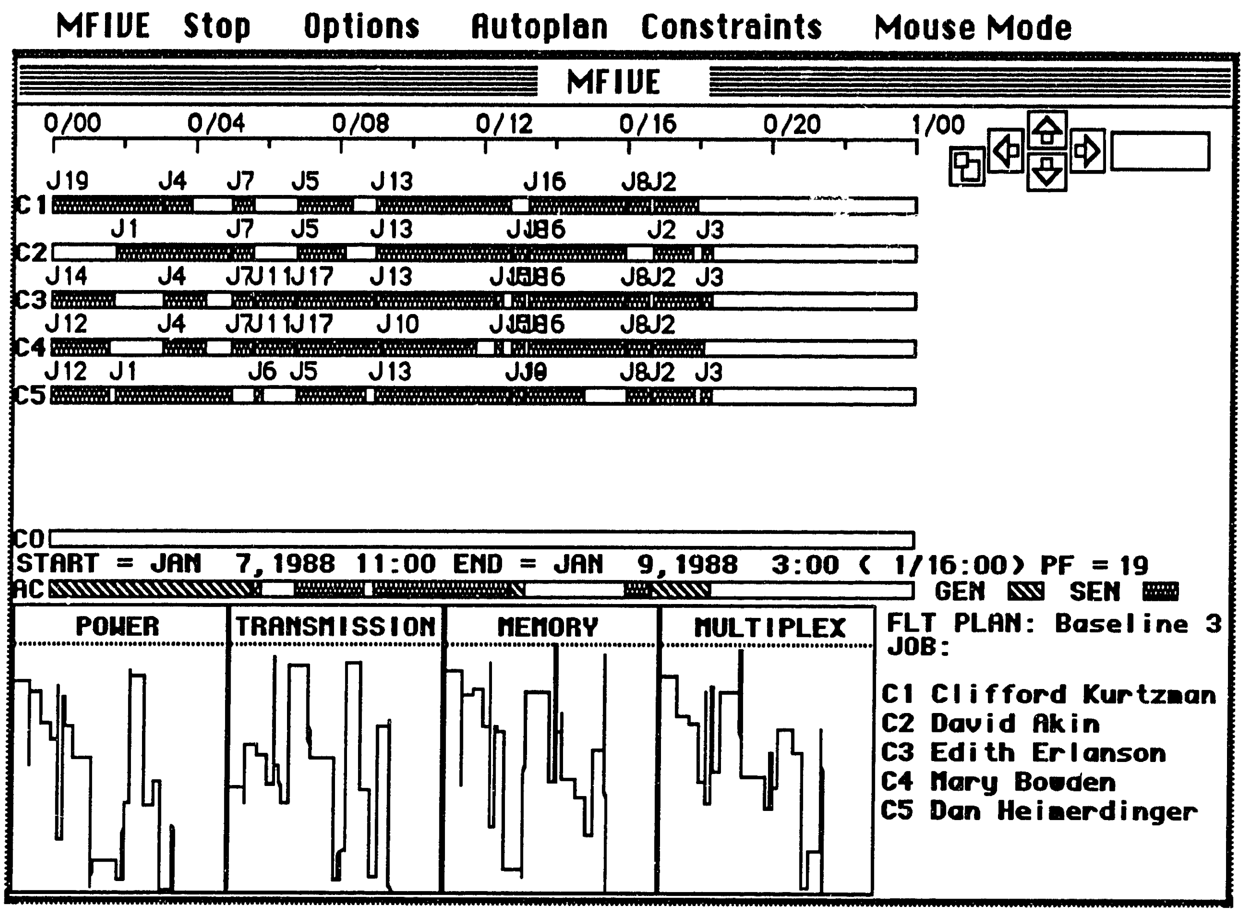


Figure A.25: Graphical Timeline for Problem 10

Results

Unlike the previous 9 problems, for this problem it is not guaranteed that by scheduling pending jobs a complete schedule will result. In fact, for this problem, a straight application of all 10 heuristics failed to produce a complete schedule. In every case, however, the iterative algorithm was able to find a complete schedule after a small number of iterations.

Table A.35: Average Number of Iterations to First Complete Solution, Problem 10

Heuristic	Deterministic	Randomized
1 SJF	15	22.333
2 LJF	6	10
3 MSLK	2	7.333
4 GRD	7.333	17
5 GRR	6.333	20.667
6 JER	5.5	15.667
7 LFT	3.667	7.667
8 EST	6	18.333
9 MCM	6.35	14
10 CCM	6.15	3.333
Average	6.433	13.633

This data is displayed in Figure A.26.

Figure A.26: Iterations to First Complete Solution, Problem 10

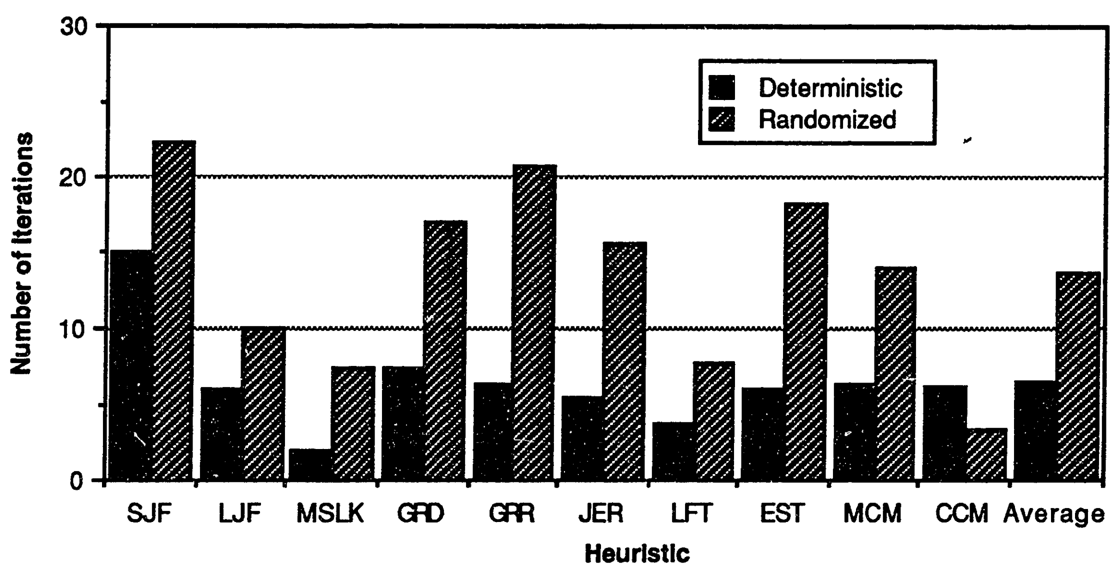


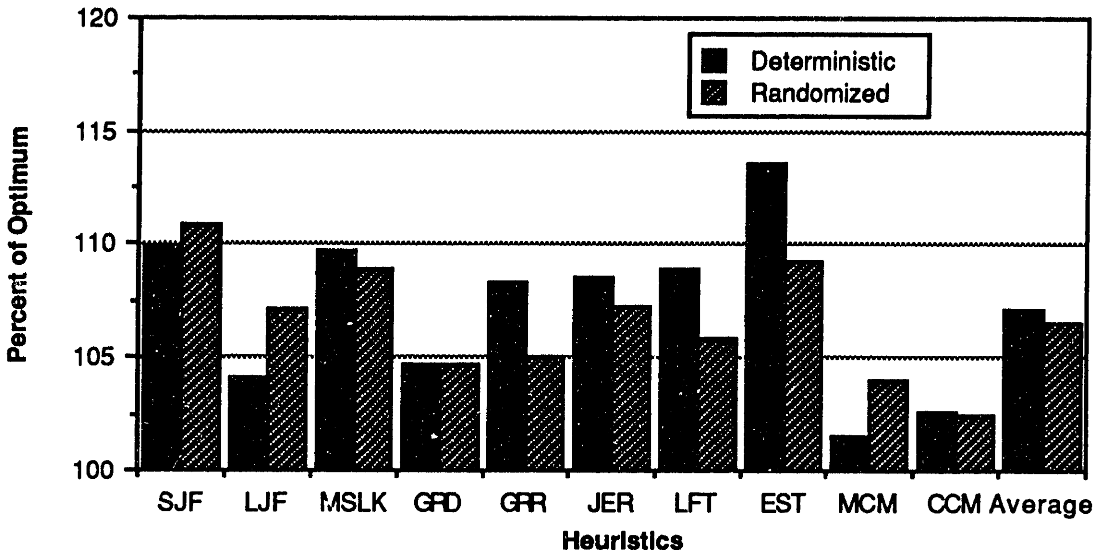
Table A.36: Average Solution after 100 Iterations, Problem 10 (Figure A.27)

Heuristic	Average After 100 Iterations: Deterministic	Randomized
1 SJF	1214	1224.333
2 LJF	1149	1182.333
3 MSLK	1211.666667	1202.667
4 GRD	1155.333333	1156
5 GRR	1195.333333	1159
6 JER	1198.5	1184.5
7 LFT	1202.666667	1168.667
8 EST	1254.333333	1206.333
9 MCM	1120.6	1147.667
10 CCM	1132.15	1130.667
Average	1183.358	1176.217

The averages in the first 2 columns are based on 20 attempts of 100 iterations each for the deterministic versions of Heuristics 9 and 10, 6 attempts for Heuristic 6 (both deterministic and randomized), and 3 attempts for each of the other heuristics.

Instead of taking 100 iterations of an algorithm, the same amount of computational work would be necessary to take 2 trial of 50 iterations each and then to take them minimum of the resulting values. This was calculated for each of the 10 heuristics applied to this problem. For the deterministic algorithms, the average was 1187.184, and for the randomized algorithms, the average was 1185.600. Both results are worse than the above results for 100 iteration trials.

Figure A.27: Final Average Solution, Problem 10



Appendix B: MFIVE Users Guide

This section contains explicit instructions for running the MFIVE Crew Activity Planner. This section is organized to direct a user in "how to" perform the different functions of the MFIVE system. This complements the material presented in Section 6, which provides a guided tour of the MFIVE system. As necessary, the figures and material in Section 6 are referenced in order to supplement the text.

The MFIVE Crew Activity Planner runs on the Apple Macintosh Plus and Macintosh SE personal computers. The current version does not run on the Macintosh II computer. Two disks are necessary to run the MFIVE software. This first disk contains the PRIME and SCHED folders, which contain the APL computer code for the MFIVE system. A second disk is needed which contains the a Macintosh System Folder and the APL*PLUS interpreter, which can be purchased from STSC, Inc., (301) 984-5000. The materials on both disks can, if desired, be moved to a hard disk drive. All that is necessary is that the PRIME and SCHED folders remain unseparated (i.e. not put in different folders).

MFIVE can be run off an AppleShare Network, with certain limitations. The APL*PLUS system is not capable of writing to the hard disk which acts as the Server, so it is not possible to transfer information between the PRIME module (the database manager) and the SCHED module (the scheduler). It is also not possible to <u>save</u> a database or schedule to the server hard disk, but it is possible to save them to disks at the users workstation, and it is possible to <u>load</u> them from the server hard disk.

MFIVE is not capable of supporting desk accessories. This is because the APL does not refresh the screen picture after it has been altered by desk accessory usage. If desk accessory usage were allowed, it would therefore cause the screen to become messed up. It is hoped that future versions of the APL interpreter will overcome this disability.

MFIVE can require a great deal of memory, especially when trying to schedule problems with in excess of 40 jobs. Therefore it is best to turn off the RAM Cache (via the Control Panel desk accessory), or set it to a very low level, like 32K, before activating MFIVE.

MFIVE will not work with the Switcher application program. MFIVE will run under the multifinder but is not able to switch to other applications while running.

B.1 The Database Management System

The PRIME module is a database management system which is used to enter and revise parameters relating to crewmembers, jobs, tools and stowage compartments. Once this database is assembled, crewmembers and jobs can be assigned to a flight plan, which will allow the scheduling of the jobs by the crewmembers. The tool searcher can also be utilized to perform searches for missing tools.

The database management system is implemented by using the Macintosh mouse to double-click on the PRIME icon, located inside the PRIME folder. The user is then presented with a screen which should look like Figure 6.3.

B.1.1 Calling Up an Information Form

Calling up a blank information form: Click the mouse on the CREW, JOB, FLIGHT PLAN, TOOL, or STOWAGE icon to call a blank information form. This can be done whether or not there is an information form already displayed on the screen, as long as there are no windows open. At this point, there are several options:

To return to the initial configuration (Figure 6.3): hit the return key.

To add a new entry (e.g., a new crewmember) to the database: type "new". The system will then ask for the name of the new entry. After the new entry is entered, an information form will be displayed for that entry. If instead of entering a new name, the user then hits the return key, no new crewmember will be added.

To call up a previously entered information form: Type the name or number of the entry. The system is capable of performing recognition of incompletely specified names. It will also ask the user to clarify ambiguities. For example, if the user calls up a blank Crewmember Information Form and then types in "Dan", and there are two Dan's known to the system, the present the user with a window showing the complete names of both Dan's and asking the user to select one of them. In the case of Crewmembers and Jobs, the system will also recognize any of the nicknames which have also been specified for the entry.

To get help: Type in "help" and the system will provide a window containing the names of all the entries known to it. The user can then select one of the entries, and the appropriate information form will be called up. If no entry is selected, the system will return to the configuration of figure 6.3.

B.1.2 Modifying an Information Form

Revising the name of an entry: By clicking on the name of an entry, the user is given a prompt at which a new name for the entry can be entered. The user will be asked to confirm that the new entry name is correct. By clicking on NO or by hitting a return key at the prompt, no revision will be entered.

Deleting an entry: Click on the entry name, and instead of entering a revision, type the word "delete". The user will be asked to confirm if the information form (and all of the information relating to it) should be deleted. Clicking on NO will cancel the deletion.

Changing the number corresponding to an entry: The number corresponding to an entry is initially provided by the system and cannot be modified.

B.1.3 Modifying Crewmember Information

Figure 6.4 shows the crewmember information form. Each of the entries on this form can be modified as follows:

Changing a crewmember's Date of Birth: Click on the currently entered date of birth (or the words **NO INFO**). The user will then be provided with a prompt in which the date of birth can be entered. The date should be entered in the format {month, day, year}. The system will recognize many different input forms. For example, December 21, 1953 could also be entered as 12/21/53, Dec 21 53, or 12 21 1953. If replacing a previously entered date, it may not be necessary to enter the complete new entry. For example, if the previously entered date was December 21, 1953, entering simply 15 will change the date to December 15, 1953, while entering 3/11 will change the date to March 11, 1953. All dates must be after January 1, 1900.

Instead of entering a date, the user can: 1) hit the return key, which will restore the date to the previously entered value; 2) type "help", which will provide an explanation of the correct format for entering dates; or 3) type "delete", which will replace the date with the statement **NO INFO**. In the present implementation of MFIVE, crewmember data of birth is not utilized by the system, and it is therefore not necessary to enter it.

Changing a crewmember's Mass or Height: Click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which the mass or height can be entered. Mass should be specified in kilograms, and must be between 45 and 114. Height should be specified in centimeters, and must be between 137 and 214 centimeters. Instead of entering the mass or height, the user can: 1) hit the return key, which will restore the mass or height to the previously entered value; 2) type "help", which will provide an explanation of the correct format for entering the mass or height; or 3) type "delete", which will replace the mass or height with the statement **NO INFO**. In the present implementation of MFIVE, crewmember mass and height are not utilized by the system, and it is therefore not necessary to enter them.

Changing a crewmember's Assignment Start and End Dates: Click on the currently entered date (or the words **NO INFO**). The user will then be provided with a prompt in which the start or end date can be entered. The date should be entered in the format {month, day, year, hour, minute}. The system will recognize many different input forms. For example, January 7, 1997 5:00 could also be entered as 1/7/97 5 0, Jan 7 97, 5:00, or 1 7 1997 5 0. The time is entered based on a 24 hour clock. If replacing a previously entered date, it may not be necessary to enter the complete new entry. For example, if the previously entered date was January 7, 1997 5:00, entering simply 15:23 will change the date to January 7, 1997 15:23, while entering 3/11 15:23 will change the date to March 11, 1997 15:23. (If adding a new end date and only the start date has been previously entered, the start date will be used to provide the defaults, etc.) All dates must be after January 1, 1900. Additionally, the start date cannot be modified to be after the end date, and the end date cannot be modified to be before the start date. Therefore if one is to move the start and end dates to times after the currently entered start date, it is necessary to change the end date first.

Instead of entering a date, the user can: 1) hit the return key, which will restore the date to the previously entered value; 2) type "help", which will provide an explanation of the correct format for entering dates; or 3) type "delete", which will replace the date with the statement **NO

INFO**. In the present implementation of MFIVE, assignment start and end dates are not utilized by any other component of the system, and it is therefore not necessary to enter them.

Changing a crewmember's Rank: Click on the currently entered value (or the word UNKNOWN). The user will then be provided with a prompt in which the crewmember's rank can be specified. The following are valid "ranks" known to the system: Unknown, Space Station Pilot, Space Station Commander, Station Specialist 1, Station Specialist 2, Station Specialist 3, and Visiting Scientist 1. At present there is no way for a user to specify a new rank category.

Instead of entering the rank, the user can: 1) hit the return key, which will restore the rank to the previously entered value; 2) type "help", which will provide a window displaying all the allowable choices, from which one can then be selected; or 3) type "delete", which will replace the rank with the statement UNKNOWN. In the present implementation of MFIVE, crewmember rank is not utilized by the system, and it is therefore not necessary to enter it.

Altering crewmember Performance Data: Clicking on the box labelled "Performance Data" on the Crewmember Information Form will display a window like that shown in Figure 6.6. This window shows the crewmember's performance times for each of the jobs entered into the system. An asterisk (*) following a performance time indicates that the performance time is the default time for that job (i.e., there has not been any explicitly entered performance time for that crewmember on that job). An asterisk with no performance time indicates that the crewmember is unrated for the job and hence incapable of performing it. If there are more jobs than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window. The ratio of the length of the box to the total length of the window is the same as the ratio of the amount of data displayed in the window to the total amount of data present.

The performance time on a particular job can be modified by clicking on the old value. The user will then be presented with a prompt requesting that a new value be entered. In addition to specifying a new value, the user can: 1) hit the return key, which will restore the performance time to the previously entered value; 2) type "help", which will provide an explanation of the correct format for entering performance times; or 3) type "delete", which will delete the crewmember's performance time for the job, thus rendering the crewmember not capable of being assigned to perform the job. When revisions to the crewmember's performance times are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes. Information presented in the crewmember performance data windows can also be accessed by the performance data windows on the job information forms.

Altering crewmember Background Information: Clicking on the box labelled "Background Information" on the Crewmember Information Form will display a window like that shown in Figure 6.5. By clicking in this window, the cursor can be positioned to allow the input of text. Text can be deleted by clicking to the end of the text and backspacing over it. The window can be moved by clicking and moving the bar at the top. The window can be resized by clicking and moving the box in the bottom right hand corner of the window. The text in the window can be scrolled by clicking on the right, left, up, and down arrows, and by clicking and moving the bars in the right and bottom margins. When revisions are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes. In the present implementation of MFIVE, crewmember background information is not utilized by any other component of the system.

Altering crewmember Nicknames: Clicking on the box labelled "Nicknames" will bring up a window allowing the input and revision of alternate names by which the crewmember may be recognized by the system. By clicking on "Add New Entry", the user will be presented with a prompt asking for a new nickname for the crewmember. (Hitting the return key will cancel the new entry.) Clicking on "Revise" and then on a previously entered nickname, the user will be presented with a prompt requesting the user to type in a revised version of the nickname. Again, the revision can be cancelled by hitting the return key.

A nickname can be deleted by clicking on the delete button and then on the name of the nickname(s) to be deleted. Nicknames which are signified to be deleted (when the nicknames window is closed) will show in inverse type font. By clicking on them again, they will be "undeleted." The "Delete All" and the "Undelete All" buttons can be used to quickly set (or remove) all of the nicknames from being deleted. Either the "Done" button or the white rectangle in the upper left hand corner can be used to close the Nicknames window, saving all changes.

The "Cancel" button can be used to close the window, ignoring all revisions, additions, and deletions which were made. If there are more nicknames than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window.

B.1.4 Modifying Job Information

Changing the Default Job Time: The default job time is the amount of time it takes to complete a job in the absence of any specific overriding information entered through the performance data window. To modify the default job time, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which the default job time can be entered. Job times are specified in minutes. Instead of entering a job time, the user can: 1) hit the return key, which will restore the default job time to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the default job time with the statement **NO INFO**.

Changing the number of Crewmembers Required: The number of crewmembers required specifies how many crewmembers it takes to perform a job. To change the number of crewmembers required, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. The number of crewmembers required must be an non-negative integer. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the default job time with the statement **NO INFO**.

The current implementation of MFIVE considers four types of linear renewable resources. At present, it is not possible for the user to add any new resource categories. The resource usages for each job are set via the job's information form, and usage limits are set via the flight plan information form.

Changing the Power Usage: The power usage level specifies how much power (in Watts) is used throughout the performance of the job. To change the power usage level, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the power usage level with the statement **NO INFO**.

Changing the High Rate Multiplex usage: The high rate multiplex usage specifies how much high rate multiplex (from onboard data recorders, in bits/second) is required throughout the performance of the job. To change the usage level, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the high rate multiplex usage with the statement **NO INFO**.

Changing the Computer Memory usage: The computer memory usage specifies how much computer memory (bytes) is required throughout the performance of the job. To change the usage level, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the computer memory usage with the statement **NO INFO**.

Changing the Data Transmission Requirement: The data transmission requirement specifies how much communications (to the ground, bits/second) is required throughout the performance of the job. To change the usage level, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the data transmission requirement with the statement **NO INFO**.

Changing a job's Audio Level: Click on the currently entered value (or the word UNKNOWN). A prompt will then be provided in which the audio level can be specified. The audio level setting allows the specification of jobs which produce or are sensitive to either noise or vibration. The scheduler will now allow the simultaneous scheduling of jobs which are noise generating and jobs which are noise sensitive. The following are valid "audio levels": Unknown, Sensitive, Neutral, and Generating.

Instead of entering the an audio level, the user can: 1) hit the return key, which will restore the audio level to the previously entered value; 2) type "help", which will provide a window displaying all the allowable choices, from which one can then be selected; or 3) type "delete", which will replace the audio level with the statement UNKNOWN.

Altering job Performance Data: Clicking on the box labelled "Performance Data" on the Job Information Form will display a window like that shown in Figure 6.8. This window shows the performance times of each of the crewmembers for this job. The default performance time for the job is shown at the top of the window. Performance times are explicitly shown for crewmembers who have been given performance times differing from the default time. An asterisk with no performance time indicates that the crewmember is unrated for the job and hence incapable of performing it. If there are more jobs than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window. The ratio of the length of the box to the total length of the window is the same as the ratio of the amount of data displayed in the window to the total amount of data present.

The performance time for a crewmember can be modified by clicking on the old value. The user will then be presented with a prompt requesting that a new value be entered. In addition to specifying a new value, the user can: 1) hit the return key, which will restore the performance time to the previously entered value; 2) type "help", which will provide an explanation of the correct format for entering performance times; or 3) type "delete", which will delete the crewmember's performance time for the job, thus rendering the crewmember not capable of being assigned to perform the job. When revisions to the job's performance times are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes. Information presented in the job performance data windows can also be accessed by the performance data windows on the crewmember information forms.

Altering the Job Description: Clicking on the box labelled "Job Description" on the Job Information Form will display a window like that shown in Figure 6.9. By clicking in this window, the cursor can be positioned to allow the input of text. Text can be deleted by clicking to the end of the text and backspacing over it. The window can be moved by clicking and moving the bar at the top. The window can be resized by clicking and moving the box in the bottom right hand corner of the window. The text in the window can be scrolled by clicking on the right, left, up, and down arrows, and by clicking and moving the bars in the right and bottom margins. When revisions are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes.

Altering job Nicknames: Clicking on the box labelled "Nicknames" will bring up a window similar to Figure 6.10, allowing the input and revision of alternate names by which the job may be recognized by the system. By clicking on "Add New Entry", the user will be presented with a prompt asking for a new alternate name for the job. (Hitting the return key will cancel the new entry.) Clicking on "Revise" and then on a previously entered nickname, the user will be presented with a prompt requesting the user to type in a revised version of the nickname. Again, the revision can be cancelled by hitting the return key.

A nickname can be deleted by clicking on the delete button and then on the name of the nickname(s) to be deleted. Nicknames which are signified to be deleted (when the nicknames window is closed) will show in inverse type font. By clicking on them again, they will be "undeleted." The "Delete All" and the "Undelete All" buttons can be used to quickly set (or remove) all of the nicknames from being deleted. Either the "Done" button or the white rectangle in the upper left hand corner can be used to close the Nicknames window, saving all changes. The "Cancel" button can be used to close the window, ignoring all revisions, additions, and deletions which were made. If there are more nicknames than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window.

Altering the Tools Needed: The tools needed to perform a job can be modified by clicking on the box labelled "Tools Needed" on the Job Information Form. This window shows the number of copies of each tool which are needed to perform this job. No number following the name of a tool indicates that it is not required for the job. At present, tool usage information is maintained for informational purposes only; the current implementation of the MFIVE scheduler does <u>not</u> check that the number of copies of a tool needed at any time exceed the quantity available. If there are more tools than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window. The ratio of the length of the box to the total length of the window is the same as the ratio of the amount of data displayed in the window to the total amount of data present.

The tools needed information can be modified by clicking on the name of a job. The user will then be presented with a prompt requesting that a new value be entered. In addition to specifying a new value, the user can: 1) hit the return key, which will restore this quantity to the previously entered value; 2) type "help", which will provide an explanation of the correct entry format ; or 3) type "delete", which will delete this entry. When revisions are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes. Information presented on a job's tools needed window can also be accessed by the job applications window on the applicable tool information form.

B.1.5 Modifying Flight Plan Information

Once job parameters and crewmembers have been entered into the system, they can be assembled into a flight plan and then scheduled. The Flight Plan Information Form (Figure 6.11) allows the user to specify an interval during which all jobs must be scheduled and limits on usage of all the resources. Time constraints on the jobs are entered in the scheduler (Section B.2.8).

Changing a flight plan's Start and End Dates: Click on the currently entered date (or the words **NO INFO**). The user will then be provided with a prompt in which the start or end date can be entered. The date should be entered in the format {month, day, year, hour, minute}. The system will recognize many different input forms. For example, January 7, 1988 11:00 could also be entered as 1/7/88 11 0, Jan 7 88,11:00, or 1 7 1988 11 0. The time is entered based on a 24 hour clock. If replacing a previously entered date, it may not be necessary to enter the complete new entry. For example, if the previously entered date was January 7, 1988 11:00, entering simply 15:23 will change the date to January 7, 1988 15:23, while entering 3/11 15:23 will change the date to March 11, 1988 15:23. (If adding a new end date and only the start date has been previously entered, the start date will be used to provide the defaults, etc.) All dates must be after January 1, 1900. Additionally, the start date cannot be modified to be after the end date, and the end date cannot be modified to be before the start date. Therefore if one is to move the start and end dates to times after the currently entered start date, it is necessary to change the end date first.

Instead of entering a date, the user can: 1) hit the return key, which will restore the date to the previously entered value; 2) type "help", which will provide an explanation of the correct format for entering dates; or 3) type "delete", which will replace the date with the statement **NO INFO**.

Changing the Maximum Power Usage limit: The power usage limit specifies how much power (in Watts) can be used at any time. To change the maximum power usage, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the power usage level with the statement **NO INFO**.

Changing the Maximum Multiplex limit: The high rate multiplex limit specifies how much high rate multiplex (from onboard data recorders, in bits/second) can be used at any time. To change the maximum level, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the high rate multiplex usage with the statement **NO INFO**.

Changing the Maximum Memory limit: The computer memory usage specifies how much computer memory (bytes) can be used at any time. To change the maximum level, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the computer memory usage with the statement **NO INFO**.

Changing the Maximum Data Transmission limit: The data transmission requirement specifies how much communications (to the ground, bits/second) can be used at any time. To change the maximum level, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which a new value can be entered. Instead of entering a new value, the user can: 1) hit the return key, which will restore this to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the data transmission requirement with the statement **NO INFO**.

Assigning Crewmembers to a Flight Plan: Crewmembers can be assigned to a flight plan by clicking on the Assigned Crewmembers box on the Flight Plan Information Form. This will bring up a window similar to that shown in Figure 6.12. By clicking on the name of a crewmember, that crewmember will be selected (or deselected) for the flight plan. Selected crewmembers are shown in inverse font. The number of selected crewmembers is shown in the righthand column of the window. No more than seven crewmembers can be selected for a flight plan. This is because the current implementation of the MFIVE scheduler only has sufficient room on the screen for displaying seven timelines.

The "Select All" and the "Clear All" buttons can be used to quickly select (or unselect) all of the crewmembers (Select All can only be used when there are seven or less crewmembers). Either the "Done" button or the white rectangle in the upper left hand corner can be used to close the window, saving all changes. The "Cancel" button can be used to close the window, ignoring all revisions. If there are more crewmembers than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window.

Assigning Jobs to a Flight Plan: Jobs can be assigned to a flight plan by clicking on the Assigned Jobs box on the Flight Plan Information Form. This will bring up a window similar to that shown in Figure 6.13. By clicking on the name of a job, that job will be selected (or deselected) for the flight plan. Selected jobs are shown in inverse font. The number of selected jobs is shown in the righthand column of the window.

The "Select All" and the "Clear All" buttons can be used to quickly select (or unselect) all of the jobs. Either the "Done" button or the white rectangle in the upper left hand corner can be used to close the window, saving all changes. The "Cancel" button can be used to close the window, ignoring all revisions. If there are more jobs than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window.

Implementing the Scheduler: By clicking on the button labelled SCHEDULE, the MFIVE Scheduler (Section B.2) will be activated, utilizing the data specified for the jobs and crewmembers assigned to the flight plan. If any resource levels are unspecified, they will the set to zero, and if any audio levels are Unknown, they will be set to neutral.

Before entering the scheduler, MFIVE checks for the occurrence of many types of infeasibilities. For example, if the resource usage on any job assigned to the flight plan exceeds the maximum usage limit for that resource, the user will be notified so that corrective action can be taken. In this case, corrective action would be to either change the resource limit of the flight plan, change the resource usage of the job, or remove the job from the flight plan. The user will also be notified if some job takes longer than the duration of the flight plan, or if there are not enough crewmembers rated to perform a job.

If no infeasibilities are detected, MFIVE will activate the scheduler after warning the user that to do so will erase any unsaved changes made to the database. For example, if a new flight plan is assembled, and the scheduler activated without saving, the new flight plan will be transferred to the scheduler. Upon returning to the database manager, however, the none of the previously made changes will be present. Saving changes to the database manager is discussed in Section B.1.8. After the changes are saved, it will be necessary to restart the database management system, as described in Section B.1.10.

B.1.6 Modifying Tool Information

Changing the Quantity in Stock: The quantity in stock cannot be directly changed. Instead, the quantity in stock is set through modifying the Default Location and Status window, as discussed below.

Changing the Effective Volume: Clicking on the currently entered value (or the words **NO INFO**) will provide the user with a prompt in which the tool's effective volume can be modified. By convention, the effective volume of a tool cannot be greater than the effective volume of any compartment in which the tool is stowed (see Default Location and Status, below). Instead of entering the effective volume, the user can: 1) hit the return key, which will restore the effective volume to the previously entered value; 2) type "help", which will provide an explanation of the correct entry format; or 3) type "delete", which will replace the effective volume with the statement **NO INFO**.

Changing the Default Location and Status: Clicking on the Default Location and Status box on the Tool Information Form will produce a window similar to Figure 6.15. This window contains information concerning where the tool is supposed to be kept (the default location) and the tool's status (i.e., whether the tool is lost, broken, operational, etc.). A copy of a tool can be deleted by clicking on its COPY #. A new copy can be added by clicking on "CREATE NEW COPY", after which the user will be prompted to enter the tools default location and status. Default location and tool status information can be modified by clicking on the previously entered information, after which the user will be asked to enter the new default location or tool status. Instead changing the current entry, the user can: 1) hit the return key, which will restore the entry to the previously entered value; 2) type "help", which will provide a list of valid entries, from which the user can select; or 3) type "delete", which will replace the entry with the designation UNKNOWN. The window can be moved by clicking and moving the bar at the top. The window can be resized by clicking and moving the box in the bottom right hand corner of the window. The text in the window can be scrolled by clicking on the right, left, up, and down arrows, and by clicking and moving the bars in the right and bottom margins. When revisions are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window.

Altering the Usage Instructions: Clicking on the box labelled "Usage Instructions" on the Tool Information Form will display a window like that shown in Figure 6.16. By clicking in this window, the cursor can be positioned to allow the input of text. Text can be deleted by clicking to the end of the text and backspacing over it. The window can be moved by clicking and moving the bar at the top. The window can be resized by clicking and moving the box in the bottom right hand corner of the window. The text in the window can be scrolled by clicking on the right, left, up, and down arrows, and by clicking and moving the bars in the right and bottom margins. When revisions are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes.

Altering the Job Applications: The job applications window maintains information regarding which jobs a tool is needed for. It can be modified by clicking on the box labelled "Job Applications" on the Tool Information Form. At the top of this window is the number of copies of the tool currently in stock. Following the name of each job is the number of copies of each tool which are needed to perform this job. No number following the name of a job indicates that it does not require this tool. At present, job application information is maintained for informational purposes only; the current implementation of the MFIVE scheduler does not check that the number of copies of a tool needed at any time exceed the quantity available. If there are more jobs than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window. The ratio of the length of the box to the total length of the window is the same as the ratio of the amount of data displayed in the window to the total amount of data present.

The job application information can be modified by clicking on the name of a job. The user will then be presented with a prompt requesting that a new value be entered. In addition to specifying a new value, the user can: 1) hit the return key, which will restore this quantity to the previously entered value; 2) type "help", which will provide an explanation of the correct entry format ; or 3) type "delete", which will delete this entry. When revisions are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes. Information presented on a tool's job applications window can also be accessed by the tool usage window on the applicable job information form.

Implementing the Tool Searcher: By clicking on the button labelled SEARCH, the MFIVE Tool Searcher (Section 6.3) will be activated. In performing a search for a tool, the Tool Searcher utilizes user entered information regarding the tools and stowage compartments, as well as internal information about the physical layout of the space station and the usage history of the tools (i.e., who last used the tool and where). In the implementation of a real time scheduling system, this information would be generated by the scheduler, but this feature is not incorporated in the present version of MFIVE.

B.1.7 Modifying Stowage Compartment Information

Changing the Effective Volume: The effective volume specifies the size of a compartment. No tool may be stowed in a compartment with a smaller effective volume. To modify this entry, click on the currently entered value (or the words **NO INFO**). The user will then be provided with a prompt in which the effective volume can be entered. Effective volume is specified in liters. Instead of entering a value for the effective volume, the user can: 1) hit the return key, which will restore the effective volume to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the effective volume with the statement **NO INFO**.

Changing the Module Number: Click on the currently entered value (or the word **NO INFO**). The user will then be provided with a prompt in which the module number (in which the stowage compartment is located) can be specified. Modules 1 and 2 are Laboratory Modules, Modules 3 and 4 are Habitation Modules, and Module 5 is a Logistics Module. At present there is no way for a user to enter information regarding a new module.

Instead of entering the module number, the user can: 1) hit the return key, which will restore the module number to the previously entered value; 2) type "help", which will provide a window displaying all the allowable choices, from which one can then be selected; or 3) type "delete", which will replace the module number with the statement **NO INFO**.

Changing the compartment's X Location: Click on the currently entered value (or the word **NO INFO**). The user will then be provided with a prompt in which the X location (from a fixed reference point inside the module in which the stowage compartment is located) can be specified. X location is measured in meters.

Instead of entering the X location, the user can: 1) hit the return key, which will restore the X location to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the X location with the statement **NO INFO**.

Changing the compartment's Theta Location: Click on the currently entered value (or the word **NO INFO**). The user will then be provided with a prompt in which the Theta location (from a fixed reference horizon inside the module in which the stowage compartment is located) can be specified. Theta location is measured in degrees.

Instead of entering the Theta location, the user can: 1) hit the return key, which will restore the Theta location to the previously entered value; 2) type "help", which will provide assistance to the user; or 3) type "delete", which will replace the Theta location with the statement **NO INFO**.

Inspecting the Default Contents of a stowage compartment: Clicking on the box labelled "Default Contents" displays a window similar to that in Figure 6.19. This window shows the tools which are supposed to be stowed in this compartment. This information cannot be edited with this window. If it is desired to inform the system that a tool is in a different compartment, the tool's information form should be accessed, and modifications made via the Default Locations and Status box.

B.1.8 Saving Changes to the Database Management System

There are two methods for saving changes to the database management system. The first is to exit the program (by clicking on the APL Exit icon) and then typing ")SAVE" followed by a carriage return. Any changes made will then be present the next time MFIVE is activated by clicking on the PRIME icon in the Macintosh finder. After clicking on the APL Exit icon, the user is put in to the APL environment, which changes the keyboard from the normal layout. To produce the characters ")SAVE", it is necessary to type a " (shift '), followed by the word save, without hitting the shift key. On a Macintosh Plus, ")SAVE" should appear. If using a Macintosh SE, however, the output on the screen will not appear as ")SAVE", because a bug in the APL interpreter does not properly load the APL font. Nonetheless, the computer will correctly recognize the input as ")SAVE", so the command should be entered this way, followed by hitting the return key. (Instead of typing ")SAVE", it is also possible to use the SAVE option from the FILE menu.)

The second method of saving data is to use the SAVE DATABASE option from the OPTIONS menu. This makes it possible to save many different databases of information. However, any information saved with this command will not be called up the next time MFIVE is activated. It will be necessary to instead use the LOAD DATABASE option from the OPTIONS menu to recall the saved information. The LOAD DATABASE and SAVE DATABASE options also work substantially slower then using)SAVE. When using LOAD DATABASE, it is important to realize that loading the database will completely replace the current database, thus causing the loss of any revisions not saved.

When using SAVE DATABASE, the user will be asked to provide a name for the database being saved. All database names must begin with the letters "PRIME ", including the space after the word "PRIME". For example, "PRIME DATABASE 7" would be a valid name, but "DATABASE 7" or even "PRIMEDATABASE 7" would not be valid names.

A database can be deleted by using the DELETE DATABASE option from the OPTIONS menu, or it can be directly removed by moving it into the trash can in the Macintosh finder.

B.1.9 Returning to the Macintosh Finder

In order to return to the finder, the user must enter the APL environment, by clicking on the APL Exit icon. In the APL environment, typing ")OFF" (or using the QUIT option from the FILE menu) will return the user to the finder. However, unless all revision are saved (see Section B.1.8) they will be lost. As with the ")SAVE" command, to produce the characters ")OFF", it is necessary to type a " (shift '), followed by the word off, without hitting the shift key. On a Macintosh Plus, ")OFF" should appear. If using a Macintosh SE, however, the output on the screen will not appear as ")OFF", because a bug in the APL interpreter does not properly load the APL font. Nonetheless, the computer will correctly recognize the input as ")OFF", so the command should be entered this way, followed by hitting the return key.

B.1.10 Restarting the Database Management System

After returning to the APL environment, it might be desired to restart MFIVE. This would be necessary if, for example, one were to add some changes to the database and then want to save them before implementing the scheduler. This can be done by typing the letter "S" (no shift key) followed by the return key.

B.1.11 In the Event of an Execution Error

The database management system has been extensively tested and should be virtually bug free, but it is possible that there are unforeseen errors or that errors in the APL interpreter will cause execution of MFIVE to terminate. This may manifest itself by producing a system error (in which case the Macintosh may have to be completely restarted) or by having control (unexpectedly) return to the APL environment. In this case, a short error message will sometimes appear. To restart MFIVE, first type a "]", which will appear on the Macintosh Plus screen as a right pointing arrow. Second, hit the return key. Finally, type an "S" followed by hitting the return key in order to restart MFIVE (see Section B.1.10).

B.2 The MFIVE Scheduler

The scheduler is used to set time and target constraints, inspect resource usage and allow manual and automatic task scheduling. There are two methods of initiating the scheduler. The first method is to assemble a flight plan in the database management system, and then activate the scheduler by clicking on the schedule button (Section B.1.5). The user will then be presented with a scheduling worksheet like that in Figure 6.20. Section 6.2.1 explains the interpretation of the scheduling worksheet. Alternately, a previously saved scheduling problem can be implemented by using the Macintosh mouse to double-click on the SCHED icon, located inside the SCHED folder. MFIVE will then present the user with a window from which a stored schedule can be selected (see Section B.2.16).

B.2.1 Selecting a Job For Scheduling

A job can be selected for scheduling by: 1) opening the DISPLAY JOB INFO window on the OPTIONS menu, and then clicking on the name of the job to be scheduled (Figure 6.28); 2) selecting the SELECT JOB option from the AUTOPLAN menu; and 3) selecting the AUTO JOB SELECT option from the AUTOPLAN menu. The last two options will automatically select a job based upon a heuristic which can be selected by choosing the SET HEURISTIC option from the AUTOPLAN menu (Figure 6.32). Automatic job selection will not occur, however, if the heuristic is selected to be either of the last two heuristics (i.e., the Maximum Compatibility Method or the Constrained Maximum Compatibility Heuristic). These two heuristics can only be used with the COMPLETE SCHEDULE options, as explained in Section B.2.4. The difference between the SELECT JOB and the AUTO JOB SELECT options is that, after the job is scheduled, the AUTO JOB SELECT option will still remain active, and another job will be automatically selected. The SELECT JOB option, on the other hand, only is active in the selection of the next job. The AUTO JOB SELECT option can be deselected by choosing it again from the AUTOPLAN menu.

B.2.2 Selecting Crewmembers and Start Time

Once a job has been selected, a crewmember or crewmembers can be assigned by: 1) picking up (with the mouse) a rectangle corresponding to a crewmembers performance of a job (as in Figure 6.29), and moving the rectangle into the desired time slot. If multiple crewmembers are required, the system will prompt the user for additional selections; 2) clicking on the crewmembers number (as highlighted in Figure 6.34) which will assign the job to that crewmember at the earliest possible start time. If multiple crewmembers are required, the system will again prompt the user for additional selections. In this case, the job will be assigned to those selected crewmembers at their earliest mutually available start time; 3) choosing SELECT CREW from the AUTOPLAN menu, in which case MFIVE will pick the crewmember(s) and start time; and 4) choosing AUTO CREW SELECT from the AUTOPLAN menu, which will have the same effect, but which will still remain active after the job is scheduled. Crewmembers will therefore be automatically chosen for subsequently selected jobs. The AUTO CREW SELECT option can be deselected by choosing it again from the AUTOPLAN menu.

B.2.3 Notes on Using the DISPLAY JOB INFO window

As mentioned in Section B.2.1, the DISPLAY JOB INFO window can be used to select a job for scheduling. It is also possible to deselect a selected job by clicking on its name. Clicking on the name of a job while another job is already selected will have the effect of switching the selected job. Unlike all the other temporary windows generated by the MFIVE system, the screen around the DISPLAY JOB INFO window remains active while the window is open. For example, it is possible, while the window is open, to manually schedule a job by picking up a rectangle corresponding the the performance of a job by a crewmember, and moving that rectangle onto the schedule. The DISPLAY JOB INFO window will temporarily disappear while the rectangle is being moved, but will reappear after the operation is completed.

The window can be moved by clicking and moving the bar at the top. The window can be resized by clicking and moving the box in the bottom right hand corner of the window. The text in the window can be scrolled by clicking on the right, left, up, and down arrows, and by clicking and moving the bars in the right and bottom margins. Due to a bug in the APL interpreter, it is possible to generate a system error by moving the DISPLAY JOB INFO window off the screen. Therefore this should be avoided.

B.2.4 Autonomous Schedule Generation

There are two methods for instructing MFIVE to generate a complete schedule. The first is to select both the AUTO CREW SELECT and the AUTO JOB SELECT options from the AUTOPLAN menu. MFIVE will then try to make a single attempt at scheduling each of the jobs, skipping any which are unschedulable. No attempt will be made to look for better solutions.

The second method is to use the COMPLETE SCHEDULE option from the autoplan menu. The user can then request that MFIVE attempt multiple trials and iterations in an attempt to find the best possible schedule. At the conclusion of this search, the user will be presented with the best schedule found. MFIVE will terminate each iteration, however, when a job is found which cannot be feasibly entered into the schedule. It makes no attempt to schedule each of the remaining jobs which are unscheduled at this point. Therefore, when working in an environment in which there are more jobs than can be feasibly scheduled, an improved solution can usually be obtained by taking the solution given from COMPLETE SCHEDULE, and then choosing AUTO CREW SELECT and AUTO JOB SELECT. This technique may have the effect of adding several more jobs into the schedule.

As fully described in Section 6.2.6, the SET PARAMETERS option on the AUTOPLAN menu can be used to specify the characteristics and depth of the search (see Figure 6.37). The COMPLETE SCHEDULE option will attempt to maximize the total priorites of the jobs successfully scheduled, breaking any ties using the rule specified on the SET PARAMETERS form. If during use of the COMPLETE SCHEDULE option, it is desired to stop before the search is complete, then selecting END SEARCH from the SEARCH menu will terminate the search and cause the best schedule up to that point to be displayed, with control returned to the user. As noted in Section B.2.1, the Maximum Compatibility Method and the Constrained Maximum Compatibility Heuristic can be used with the COMPLETE SCHEDULE option. The user should be aware, however, that in the current implementation of MFIVE, each time COMPLETE SCHEDULE is selected (with these heuristics) the compatibility matrix must be recomputed (Section 5.1.3), and this may require considerable computer time, especially if there are over 20 jobs (see Section 5.3.1).

B.2.5 Changing the Timeline Origin and Scale

The timeline origin and scale can be modified by clicking on the arrows in the upper right of the scheduling worksheet. This process is completely described in Section 6.2.1. The scale cannot be set to less than 1 minute per pixel (or 6 hours total duration).

B.2.6 Inspecting the Resource Levels

The use of the resource windows is described in Section 6.2.1. The window which is displayed by clicking on one of the resource boxes can be moved by clicking and moving the bar at the top. The window can be resized by clicking and moving the box in the bottom right hand corner of the window. The window is closed by clicking on the white box in the upper left hand corner of the window. While the window is open, no other changes to the scheduling worksheet can be made.

B.2.7 Setting Job Priorities

Job priority information can be altered by selecting JOB PRIORITES from the CONSTRAINTS menu (Section 6.2.7). When scheduling begins, each of the jobs is assigned a priority of 1, but this can be manually raised or lowered if a job is deemed to be more or less important. It may also be desirable to raise or lower the priorities of jobs which are observed to cause difficulty in scheduling (using the COMPLETE SCHEDULE option).

On the priority window (Figure 6.39) is the name of each job followed by its priority. If there are more jobs than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window. The ratio of the length of the box to the total length of the window is the same as the ratio of the amount of data displayed in the window to the total amount of data present.

The job priorities can be modified by clicking on the name of a job. The user will then be presented with a prompt requesting that a new value be entered. In addition to specifying a new value, the user can: 1) hit the return key, which will restore this quantity to the previously entered value; 2) type "help", which will provide an explanation of the correct entry format ; or 3) type "delete", which will delete this entry. When revisions are completed, the window can be closed and the revisions made permanent by clicking on the white box in the upper left hand corner of the window. If it is decided that the revisions should not be kept, then clicking on the grey box in the upper right hand corner of the window will close the window while cancelling any changes.

B.2.8 Setting Time Constraints

The process of setting time constraints is discussed in Section 6.2.8. The time constraints window can be moved by clicking and moving the bar at the top. The window can be resized by clicking and moving the box in the bottom right hand corner of the window. The text in the window can be scrolled by clicking on the right, left, up, and down arrows, and by clicking and moving the bars in the right and bottom margins. When revisions are completed, the window can be closed by clicking on the white box in the upper left hand corner of the window.

Time constraints are entered in the form {days, hours, minutes} from the start of the flight plan. The system will recognize many different input forms. For example, 1/15:25 could also be entered as 1 15 25. The time is entered based on a 24 hour clock. If replacing a previously entered day and time, it may not be necessary to enter the complete new entry. For example, if the previously entered time was 1/15:25, entering simply 30 will change the entry to 1/15:30, while entering 11:45 will change the entry to 1/11:45.

Instead of entering a day and time, the user can: 1) hit the return key, which will restore the day and time to the previously entered value; or 2) type "help", which will provide an explanation of the correct format for entering the day and time.

B.2.9 Setting Target Constraints

The process of setting target constraints is discussed in Section 6.2.9. After selecting the TARGETS option from the CONSTRAINTS window, the user is presented with a window like that in Figure 6.45, from which a job can be chosen by clicking on its name. Jobs with an asterisk next to their name are already scheduled and cannot have their priorities changed. If there are more jobs than can fit onto the screen, the window can be scrolled by clicking on the right or left arrows or by clicking on and moving the box at the bottom of the window. Clicking on the white rectangle in the upper left hand corner of the window will cause the addition of target constraints to be cancelled.

Once a job has been selected, a target constraint window like that in Figure 6.46 will appear. A previously entered time constraint can be deleted by clicking on its number beneath the word DEL. The start and end times of target constraints can be changed by clicking on the old values and then entering new values at the prompt. The start time of a constraint must be before the end time, so if it is desired to modify a window to start (and end) after the current end time, it will be necessary to modify the end time first. If there are more than 2 target constraints entered, the window can be scrolled by clicking on the up and down arrows located under the word SCROLL. A new target constraint can be entered by clicking on the start time area on the first line that does not contain a constraint.

As with the time constraints, target constraints are entered in the form {days, hours, minutes} from the start of the flight plan. The system will recognize many different input forms. For example, 1/15:25 could also be entered as 1 15 25. The time is entered based on a 24 hour clock. If replacing a previously entered day and time, it may not be necessary to enter the complete new entry. For example, if the previously entered time was 1/15:25, entering simply 30 will change the entry to 1/15:30, while entering 11:45 will change the entry to 1/11:45. (If adding a new end day and time and only the start day and time has been previously entered, the start day and time will be used to provide the defaults, etc.)

Instead of entering a day and time, the user can: 1) hit the return key, which will restore the day and time to the previously entered value; or 2) type "help", which will provide an explanation of the correct format for entering the day and time.

Once all modifications to the target window are completed, they can be saved by clicking on the word DONE. Clicking on the word CANCEL will erase any changes.

B.2.10 Inspecting a Schedule

With the MOUSE MODE menu set at IDENTIFY JOB (Figure 6.48), clicking on a job will show a window displaying the name of the job and its status (Figure 6.22). When a job has been selected for scheduling, clicking on a blacked out region will produce a window displaying the reason the region is blacked out (Figure 6.30 and Figure 6.47). This will occur regardless of the setting of the MOUSE MODE menu.

B.2.11 Modifying the Status of a Job

If the assignment of a job to a crewmember is designated as FIXED, using the COMPLETE SCHEDULE option will not change the assignment. On the other hand, if the assignment of a job to a crewmember is designated as FREE, the COMPLETE SCHEDULE option will be free to assign the job to any crewmember(s). If a job requires multiple crewmembers, then it is possible for only some of the assignments to be fixed and the others to be free. The crewmember assignments of jobs which are manually assigned to crewmembers, by either directly moving the rectangles or by clicking on the crewmembers numbers (see Section B.2.2), are initially designated as FIXED. The crewmember assignments of jobs which are automatically scheduled by using AUTO CREW SELECT, SELECT CREW, or COMPLETE SCHEDULE are initially designated as FREE. Setting the MOUSE MODE menu to FREE OR FIX ASSIGNMENT and then clicking on a job on a crewmember's schedule will toggle the crewmember's assignment on that job from FIXED to FREE, or from FREE to FIXED. This method of changing a job's status, as well as all those presented below, will only work when there is no job currently selected for scheduling. If a job is selected for scheduling, the mouse will continue to work only in the IDENTIFY JOB mode.

If the start time of a job is designated as FIXED, using the COMPLETE SCHEDULE option will not change the start time. On the other hand, if the start time of a job is designated as FREE, the COMPLETE SCHEDULE option will be free to start the job at any time (within the limitations of the other constraints). The start times of jobs which are manually assigned to crewmembers by directly moving the rectangles (see Section B.2.2), are initially designated as FIXED. The crewmember assignments of jobs which are scheduled by clicking on the crewmember number or automatically scheduled by using AUTO CREW SELECT, SELECT CREW, or COMPLETE SCHEDULE are initially designated as FREE. Setting the MOUSE MODE menu to FREE OR FIX START TIME and then clicking on a job on the schedule will toggle the start time of that job from FIXED to FREE, or from FREE to FIXED.

Choosing the FIX JOB option from the MOUSE MODE menu and then clicking on a job on the schedule will set both the job's assignment and start time to FIXED. Similarly, choosing the FREE JOB option from the MOUSE MODE menu and then clicking on a job on the schedule will set both the job's assignment and start time to FREE.

By setting the MOUSE MODE to RESCHEDULE NUMERICALLY, and clicking on a job in the schedule, a window like that in Figure 6.49 will appear. Each crewmember has a box under the column labelled "ASSIGNED?" which shows the assignment status of each crewmember on the job. If the box is empty, then the crewmember is not assigned the job. If the box is grey, then the assignment is FREE. If the box is black, then the assignment is FIXED. By clicking on the boxes it is possible to change them from white to black, black to white, grey to black (i.e., FREE to FIXED), or black to grey. Clicking on the ENTER button will then reschedule the job, utilizing these changes (if the start time is free, it may be changed by the scheduler). Of course, if more crewmembers are chosen (set to black) than are needed to perform the job, then the user will be notified that the selection is not possible. Similarly, if a crewmember is chosen to perform the job who is already busy (and the start time is fixed) then the user will be told that there is no feasible scheduling.

Two columns in the window in Figure 6.49 are labelled TOTAL WORKLOAD and FIRM WORKLOAD. The total workload shows the total number of minutes to which the crewmember is assigned jobs in current schedule. The firm workload shows the total number of minutes to which the crewmember is required assignment because of having a FIXED assignment. These assignments will not be altered by using the COMPLETE SCHEDULE option.

If the start time is FREE, it will appear in normal font (as in Figure 6.49). If the start time is FIXED, it will appear in inverse font. By clicking on the time and then explicitly entering a new time (or even the same time), the time will be designated FIXED. If the start time is already FIXED, it can be changed to FREE by typing "delete" instead of entering a new time. Clicking on the ENTER button will then reschedule the job, utilizing this change (if the crewmember assignments are free, they may be changed by the scheduler). If the start time is changed to a time at which it is not possible to schedule the job, then the user will be so notified. It is possible to modify both the start time and the crewmember assignment status before clicking on ENTER.

The difference between using the ENTER button and using the RESCHEDULE button is that using ENTER will effect changes only to the job being modified. RESCHEDULE attempt a complete replan of the schedule using the same type of search as does COMPLETE SCHEDULE (utilizing the heuristic designated with the SET HEURISTIC option and the parameters designated with the SET PARAMETERS option). Only those job assignments and start times in the current schedule designated as FIXED will be necessarily retained.

The final way to change the status of a job is to set the MOUSE MODE menu to the RESCHEDULE VISUALLY option. Then clicking on a job on the schedule and moving the rectangle to a new time will FIX that crewmember assignment and start time, and also initiate a rescheduling utilizing COMPLETE SCHEDULE.

B.2.12 Removing a Job from the Schedule

A job can be removed from the schedule by setting the MOUSE MODE to RESCHEDULE NUMERICALLY and then clicking on the job. A window will appear similar to that in Figure 6.49. Clicking on UNSCHEDULE will cause the job to be removed from the current schedule. This method of unscheduling a job will only work when there is no job currently selected for scheduling. If a job is selected for scheduling, the mouse will continue to work only in the IDENTIFY JOB mode.

B.2.13 Resetting the Schedule

Selecting the UNSCHEDULE ALL option from the AUTOPLAN menu will cause all of the scheduled jobs to become unscheduled. Selecting the RESTART SCHEDULE option from the AUTOPLAN menu will, in addition to unscheduling all of the jobs, erase all of the time and target constraints.

B.2.14 Redrawing the Screen

If for some reason the screen should become messed up, choosing the REDRAW SCREEN option from the OPTIONS menu will cause the screen to be redrawn.

B.2.15 Printing Schedules

Selecting PRINT SCHEDULE from the OPTIONS menu can be used to obtain a printout of a schedule. First a window will be displayed asking the user to designate if a printout is desired which lists the schedule by crewmember (showing each job, and start and end time, to which the crewmember is assigned), by job (showing which crewmember(s) are assigned to perform each job), or by picture (showing a representation of the schedule similar to that which appears on the Macintosh screen when running MFIVE). With pictorial output, the temporal width of each segment of the timeline can be specified by the user (each minute/pixel is equivalent to 6 hours of the schedule included in each picture). Due to a bug in the APL interpreter, pictorial output will not work with the laserwriter as the output device. It does, however, work with an imagewriter. The best way to get pictorial output with the laserwriter is to get the Macintosh screen to the desired state and then type the character "Command-Shift-3", which will create a MacPaint file containing an image of the screen. Each successive image created in this manner will be stored in a file labelled SCREEN 0, SCREEN 1, as so on, up to a maximum of SCREEN 9. When the user returns to the finder after using MFIVE, these files can be edited or printed using MacPaint or some other graphics program.

B.2.16 Saving, Loading and Deleting Schedules

At any point in the scheduling process, it is possible to save a schedule, including all time and target constraints. This is done by selecting the SAVE SCHEDULE option from the OPTIONS menu. The user will be asked to provide a name for the schedule being saved. All schedule names must begin with the letters "SCHED ", including the space after the word "SCHED". For example, "SCHED PROBLEM 3" would be a valid name, but "PROBLEM 3" or even "SCHEDPROBLEM 3" would not be valid names.

A previously stored schedule can be loaded into memory by using the LOAD SCHEDULE option from the OPTIONS menu. Loading a schedule into memory will completely replace the scheduling problem that is in memory before the stored schedule is loaded.

A schedule can be deleted by using the DELETE SCHEDULE option from the OPTIONS menu, or it can be directly removed by moving it into the trash can in the Macintosh finder.

B.2.17 Returning to the Macintosh Finder

In order to return to the finder, the user must enter the APL environment, by selecting the RETURN TO APL option from the OPTIONS menu. In the APL environment, typing ")OFF" (or using the QUIT option from the FILE menu) will return the user to the finder. To produce the characters ")OFF", it is necessary to type a " (shift '), followed by the word off, without hitting the shift key. On a Macintosh Plus, ")OFF" should appear. If using a Macintosh SE, however, the output on the screen will not appear as ")OFF", because a bug in the APL interpreter does not properly load the APL font. Nonetheless, the computer will correctly recognize the input as ")OFF", so the command should be entered this way, followed by hitting the return key.

B.2.18 Restarting the Scheduler

After returning to the APL environment, it might be desired to restart MFIVE. This can be done by typing the letter "SL" (no shift key) followed by the return key. The user will then be asked to select a schedule to be loaded.

B.2.19 In the Event of an Execution Error

The scheduler has been extensively tested and should be virtually bug free, but it is possible that there are unforeseen errors or that errors in the APL interpreter will cause execution of MFIVE to terminate. This may manifest itself by producing a system error (in which case the Macintosh may have to be completely restarted) or by having control (unexpectedly) return to the APL environment. In this case, a short error message will sometimes appear.

In order to restart the scheduler, the first step is to type a "]", which will appear on the Macintosh Plus screen as a right pointing arrow. Next, hit the return key. Finally, type "SL" followed by hitting the return key to restart MFIVE (see Section B.2.18). This method will unfortunately cause any changes made to the schedule since it was last saved (if it was ever saved) to be lost.

Appendix C: MFIVE Programmers Guide

The MFIVE programming environment is organized into two APL workspaces. The PRIME workspace contains the software for the database management system and the tool searcher. The SCHED workspace contains the software for the activity scheduler.

This section will provide an overview of the organization of the workspaces, a description of each of the global variables and functions (programs) in the workspaces, and finally a listing of the APL code of each of the functions. This section assumes some knowledge of APL programming. Details of the particular APL system which was used can be found in [STSC, 1986].

C.1 The PRIME Workspace

When the PRIME workspace is loaded, the latent expression 'S' is activated, starting execution of the function S. S is a niladic function (it has no arguments) and returns no explicit result. This function directs erasure of the screen, draws the word MFIVE in the bar at the top of the screen, and calls the icon manager (the function ICON). The icon manager is a generic function which will display a set of icons at the top of the screen and takes action if one of the icons is clicked using the Macintosh mouse. The icon manager also oversees the process of modifying the information forms and using the OPTIONS menu.

C.1.1 The ICON Manager

ICON is a monadic function (it has one argument, ICONNAMS) and returns no explicit result. ICON takes as input a character vector with the names of each of the icons separated by spaces. For example, the first icon name used by the function S in calling ICON is "XCREW". This serves as a pointer to the function ICON as to where information regarding the XCREW icon is located. For example, the variable which is created by adding the letters FORM to the icon name (XCREWFORM) contains a picture of the information form relating to that icon name (in this case, the Crewmember Information Form). Similarly, adding the letters FORMRECTS (XCREWFORMRECTS) gives the name of the variable indicating the regions on the information form which are sensitive to being clicked. Adding the letters FORMSIZE (XCREWFORMSIZE) give the name of the variable indicating the size of the information form. Adding the letters

ICON (XCREWICON) gives the name of the variable containing the picture of the of the icon. Adding the letters ICONEXE (XCREWICONEXE) gives the name of the variable containing the action (an executable expression) to be taken by MFIVE when the icon is clicked on. The executable expression directs the icon manager to produce a blank information form using the function FORMGET. FORMGET is a dyadic function (i.e., it has two arguments, A and D) and produces no explicit result. The variable A contains a character vector with the prefix of the form (i.e., XCREW) and the variable D is null (for producing a blank information form; if the form is to be filled in by data pertaining to a specific entry, then D is a scalar containing the object number of the entry to be filled in, as described in Section C.1.2).

Adding the letters OPTS (XCREWOPTS) gives the name of a variable which indicates the actions to be taken whenever any of the sensitive regions on the information form (XCREWFORMRECTS) are clicked. Adding the letters FORMFILL (XCREWFORMFILL) gives the name of the function which is used to fill in the data in the information form. Data pertaining to the JOB icon is found with the pointer XJOB, data pertaining to the FLIGHT PLAN icon is found with the pointer XMAN, data pertaining to the TOOL icon is found with the pointer XTOOL , and data pertaining to the STOWAGE icon is found with the pointer XSTOW. The APL Exit icon does not call up an information form. The picture of the icon is found in the variable APLICON, and the action taken when the icon is clicked (i.e., return to APL) is found in the variable APLICONEXE.

Summary of variables relating to icons and information forms:

CREW	JOB	FLIGHT PLAN	TOOL	STOWAGE
XCREWFORM	XJOBFORM	XMANFORM	XTOOLFORM	XSTOWFORM
XCREWFORMRECTS	XJOBFORMRECTS	XMANFORMRECTS	XTOOLFORMRECTS	XSTOWFORMRECTS
XCREWFORMSIZE	XJOBFORMSIZE	XMANFORMSIZE	XTOOLFORMSIZE	XSTOWFORMSIZE
XCREWICON	XJOBICON	XMANICON	XTOOLICON	XSTOWICON
XCREWICONEXE	XJOBICONEXE	XMANICONEXE	XTOOLICONEXE	XSTOWICONEXE
XCREWOPTS	XJOBOPTS	XMANOPTS	XTOOLOPTS	XSTOWOPTS

And the following functions:

XCREWFORMFILL	XJOBFORMFILL	XMANFORMFILL	XTOOLFORMFILL	XSTOWFORMFILL

Each of these functions is monadic and returns no explicit result. The single argument (represented by the variable A) of these functions is the object number (see Section C.1.2) of the information form which is to be filled in.

C.1.2 Functions for Modifying Information Form Names and Nicknames

The names (and nicknames) of each of the items represented on the information forms are stored in variables with names that end with the suffix LIST: CREWLIST, JOBLIST, MANLIST, TOOLLIST, and STOWLIST. For example, the variable CREWLIST is a matrix which contains (starting at the second row) the names (one to a row) of the crewmembers. The variable CREWN (or alternately JOBN, TOOLN, MANN, STOWN) is a matrix which contains the nicknames of crewmembers. The variable CREWNN (or alternately JOBNN, TOOLNN, MANNN, STOWNN) is a vector which contains the inventory numbers of each of the crewmembers and the nicknames. This is the number that appears beneath the name of the crewmember on the information form. The total length of the vector CREWNN is thus equal to one less than the total number of rows in the matrices CREWLIST and CREWNN. The object number of a crewmember is the crewmember's row number within CREWLIST minus 1 (or simply its position in the vector CREWNN). For example, to find out which crewmember is crewmember 5 (inventory number), one would find the first position within CREWNN that there is a 5 (the object number of Crewmember 5). (Any subsequent occurrences of the number 5 within CREWNN would correspond to nicknames for Crewmember 5 stored in CREWN.) Suppose that there is a 5 in the eighth position within the CREWNN vectors. Then the name in the 9th row of CREWLIST corresponds to the crewmember with inventory number 5. The inventory number of a crewmember will never change, although it is possible for the object number to change. For example, if the crewmember with object number 4 is deleted from the database, then all crewmembers with object numbers greater than 4 will have their object numbers diminished by 1.

The monadic function GETNAME is called by FORMGET to add a new entry (e.g. a new crewmember) to the database. GETNAME takes the single argument A, which contains the prefix of the form being filled (XCREW, XJOB, XMAN, XTOOL, or XSTOW).

The dyadic function REVISENAME allows revision of an information form name or deletion of the entire form. REVISENAME has 2 arguments, A and G. The variable A is a character vector which contains the prefix of the form being modified (XCREW, XJOB, XMAN, XTOOL, or XSTOW), while G is a 4 element numeric vector which contains the coordinates of the region on the screen (upper left and lower right) in which the name of the entry is to be displayed. The variable REVISENAMEHELP contains a help message.

The dyadic function REVISENICKNAME is used to revise crewmember and job nicknames. REVISENICKNAME accepts 2 arguments, B and OBJ. B contains the prefix of the type of form being modified (XCREW, XJOB, XMAN, XTOOL, or XSTOW), while OBJ is the object number of the crewmember or job whose nicknames are being modified.

C.1.3 Revising the Information Forms

Functions with names which begin with the prefixes CREW, JOB, MAN, TOOL or STOW, and end with the suffix GET are used to directs the filling in of slots on the information forms. For example, the monadic function CREWHTGET directs the process by which the height of each crewmember (on the Crewmember Information Form) can be modified. The argument of the function CREWHTGET is the variable OBJ, which is the object number of the crewmember whose height is being modified. The variable CREWHT is a vector which contains the heights of each of the crewmembers, ordered by object number. The variable CREWHTDEF contains the default value for the height: if the value stored for a crewmember in CREWHT is equal to CREWHTDEF, then that crewmember's information form will read **NO INFO**.

Variables with the prefix XCREW, XJOB, XMAN, XTOOL, or XSTOW, and with the suffix HELP, contain the help message for the similarly named functions which utilized them. For example, the variable XCREWHTHELP contains the help message for the function CREWHTGET. Table C.1 lists the functions and variables that are used to fill in the information forms. All of the functions are monadic with argument OBJ (the object number), except for CREWASGET and MANASGET, which fill in the crewmember start and end times and the flight plan start and end times. These two functions are dyadic, with arguments A and OBJ. The variable A is 1 if a start time is requested, or 2 if an end time is requested.

A few of the variables in Table C.1 have a non-standard format. The variables PERFDATA1 and PERFDATA2 are vectors which contain the performance times of each of the jobs. A negative entry in PERFDATA1 signifies the inventory iumber of a job, and the value at the corresponding position in PERFDATA2 is the default performance time for that job. The inventory numbers of all crewmembers which have performance times different than the default time are listed as positive entries in PERFDATA1 following the negative entry corresponding to the job (until the next negative entry in PERFDATA1). The corresponding entries at the same positions in PERFDATA2 correspond to the performance times of those crewmembers.

Table C.1: Functions and Variables for Filling in the Information Forms

Function Name	Data Variable(s)	Default Variable	Help Variable	Purpose
CREWASGET	CREWAS	CREWASDEF	XCREWASHELP	Crew Start and End Dates
CREWBDGET	CREWBD	CREWBDDEF	XCREWBDHELP	Crewmember Date of Birth
CREWHTGET	CREWHT	CREWHTDEF	XCREWHTHELP	Crewmember Height
CREWINFOGET	*	None	None	Crew Background Info
CREWMSGET	CREWMS	CREWMSDEF	XCREWMSHELP	Crewmember Mass
CREWPERFGET	PERFDATA1 PERFDATA2	JOBTIMEDEF	XCREWPERFHELP	Crew Performance Times
CREWRANKGET	CREWRANK	CREWRANKDEF	None	Crewmember Rank
JOBAUDIOGET	JOBAUDIO	JOBAUDIODEF	None	Job Audio Status
JOBDEFGET	PERFDATA1 PERFDATA2	JOBTIMEDEF	XJOBDEFHELP	Job Default Time
JOBINFOGET	*	None	None	Job Description
JOBMEMORYGET	JOBMEMORY	JOBMEMORYDEF	XJOBMEMORYHELP	Job Memory Usage
JOBMULTIPLEXGET	JOBMULTIPLEX	JOBMULTIPLEXDEF	XJOBMULTIPLEXHELP	Job Multiplex Usage
JOBPERFGET	PERFDATA1 PERFDATA2	JOBTIMEDEF	XJOBPERFHELP	Job Performance Times
JOBPOWERGET	JOBPOWER	JOBPOWERDEF	XJOBPOWERHELP	Job Power Usage
JOBSIZEGET	JOBSIZE	JOBSIZEDEF	XJOBSIZEHELP	Crewmembers Required
JOBTOOLGET	TOOLUSE1 TOOLUSE2	TOOLUSEDEF	XJOBTOOLHELP	Tool Usage
JOBTRANSGET	JOBTRANS	JOBTRANSDEF	XJOBTRANSHELP	Job Transmission Usage
MANASGET	MANAS	MANASDEF	XMANASHELP	Flight Plan Start & End
MANCREWGET	MANCREW	None	None	Flight Plan Crewmembers
MANJOBGET	MANJOB	None	None	Flight Plan Jobs
MANMEMORYGET	MANMEMORY	MANMEMORYDEF	XMANMEMORYHELP	Flight Plan Memory
MANMULTIPLEXGET	MANMULTIPLEX	MANMULTIPLEXDEF	XMANMULTIPLEXHELP	Flight Plan Multiplex
MANPOWERGET	MANPOWER	MANPOWERDEF	XMANPOWERHELP	Flight Plan Power
MANTRANSGET	MANTRANS	MANTRANSDEF	XMANTRANSHELP	Flight Plan Transmission
STOWMODGET	STOWMOD	STOWMODDEF	None	Stowage Module Location
STOWTHETAGET	STOWTHETA	STOWXYZDEF	XSTOWTHETAHELP	Stowage Theta Location
STOWTOOLGET	TOOLSTOW	None	None	Stowage Contents
STOWVOLGET	STOWVOL	None	XSTOWVOLHELP	Stowage Volume
STOWXGET	STOWX	STOWXYZDEF	XSTOWXHELP	Stowage X Location
TOOLINFOGET	*	None	None	Tool Usage Instructions
TOOLSTOWGET	TOOLSTOW TOOLSTAT	None	None	Tool Location and Status
TOOLUSEGET	TOOLUSE1 TOOLUSE2	TOOLUSEDEF	XTOOLUSEHELP	Tool Job Applications
TOOLVOLGET	TOOLVOL	TOOLVOLDEF	XTOOLVOLHELP	Tool Volume

*Data for CREWINFOGET, JOBINFOGET, and TOOLINFOGET is stored in variables of the form CREW[]INFO, JOB[]INFO, and TOOL[]INFO, where the brackets are replaced by the inventory number of the crewmember, job, or tool.

The variables TOOLUSE1 and TOOLUSE2 are similarly organized, with negative entries in TOOLUSE1 corresponding to inventory numbers of the tools, and positive entries corresponding to inventory numbers of the job which use the tools. The entries in TOOLUSE2 corresponding to the tools signify the quantity of the tool in stock, and the entries in TOOLUSE2 corresponding to the jobs signify the number of copies of the tool needed to perform the job.

The variable TOOLSTOW (and TOOLSTAT) contains the negative of the inventory numbers of each of the tools, followed by one positive integer for each copy of the tool in stock. These positive integers correspond to the inventory numbers of the stowage location (or tool status) of each copy of the tool.

The variable MANCREW (and MANJOB) contains the negative of the inventory numbers of each of the flight plans. Each negative number is followed by positive integers corresponding to the inventory number of each crewmember (or job) assigned to the flight plan.

C.1.4 Other Global Variables in the PRIME Workspace

The variables ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, and ELEVEN all contain the scalar integer indicated by their name. The variables NONE, NTWO, NFOUR, and NFIVE contain the integers -1, -2, -4, and -5 respectively. The variable PFIVE contains the number .5. The variable IZ contains a vector of zero length.

The variable AV contains the APL character set. ALFL contains the lower case alphabet and ALFU contains the upper case alphabet. AV125 contains the 125 character in AV, which is the "|" character. This character is used as a cursor and to separate fields in windows produced by MFIVE.

The variable ACTIONSTOP contains instructions on what should happen when program interruption occurs. When ACTIONSTOP is null, control will return to the APL environment.

The variable BEEP specifies the tone heard whenever MFIVE produces a beep sound.

The variable MONTHS is a twelve row character matrix with each row containing the first three letters of the name of one of the twelve months of the year. The variable MOS2 is a twelve element numeric vector containing the number of days in each month in a non-leap year. The variable MOS is a 24 x 1 column matrix. The first twelve rows contain the cumulative number of days through the end of each month in a non-leap year. The second twelve rows contain the cumulative number of days through the end of each month in a leap year.

The variables BINH1, BINH2, BINH3, BINH4, BINH5, BINH6, and BINH7 contain text used to create the output window used by the tool searcher. The variables H1STOW, H1TOOL, H2TOOL, and H3TOOL contain text which is used by the stowage contents window and the default location and status windows.

The variables BLACKPAT, GREYPAT, and WHITEPAT contain the pen patterns for producing black, grey, and white drawings on the screen.

The variable COMLIST contains a matrix with the names of each of the forms (XCREW, XJOB, XMAN, XTOOL, and XSTOW).

The variables CORNER, RIGHTARROW, LEFTARROW, UPARROW, and DOWNARROW contain the pictures for components of the windows which MFIVE draws.

CREWRANKLIST contains the names of the ranks which can be selected. CREWRANKN is a null matrix (there are no nicknames for the ranks, but if there were, they would be stored in this matrix). CREWRANKNN the inventory numbers of the ranks. Similarly, JOBAUDIOLIST contains the names of each of the audio levels, JOBAUDION is a null matrix, and JOBAUDIONN contains the inventory numbers of the audio levels. STATLIST contains a list of all the possible tool status', STATN contains null matrix, and STATNN contains the tool status inventory numbers.

The variable DESKTOP contains the message that MFIVE does not support desk accessories. The variable EXITMSG contains a message reminding the user to save the workspace before quitting. The variable NHA contains the message "No help is currently available."

The variable HSCROLL contains a number indicating how far a window should be scrolled when clicking on the right and left arrows.

The variable LOGO contain a representation of the SSL logo that is presented when MFIVE is first entered. The variable MFIVEICON2 contains a representation of the MFIVE logo. The variable MFIVEHEADER contains the information necessary to draw the word MFIVE in the bar at the top of the screen.

The variable NULL contains a list of "null" words which are ignored by the language interpretation software in the function GETINP (see Section C.1.7).

The variable LOADRECT contains the size of the window presented when loading, saving, or deleting a database. The variable SCREEN contains the size of the Macintosh screen that is used by MFIVE.

The variable OPTIONS contains the text for the OPTIONS menu. The variable OPTIONSN contains the menu number for the OPTIONS menu.

The variable MODLENGTH contains the length of the space station modules. The variable MODNAMES contains the names of the stowage modules. The variable MODMATRIX contains the module location and type of each module. The variable SEARCHCMATRIX is a 0-1 three dimensional matrix which describes the connectivity of the modules. It consists of a plane of ones and zeros for each module. Each plane is indexed along both axes by the module numbers. A 1 at the intersection of two module numbers indicates that the module whose connectivity is described in this plane lies on a path between the two indexing modules.

The variable TOOLUSERS contains the history of who has used each tool. The negative of the inventory number of each tool is followed by the inventory numbers of the crewmembers who used it last. The variable TOOLUSEMODS similarly contains the modules in which the tool was last used, and the variable TOOLUSEX contains the X location (in each of these modules) in which the tools were last used. The variable TOOLHIST contains the inventory numbers of the stowage compartments in which the tool has previously been found.

The variable PRIMEDISK contains the name of the folder (or disk, if it is not in a folder), in which the PRIME workspace is supposed to be located. The variable SCHEDDISK contains the name of the folder (or disk, if it is not in a folder), in which the SCHED workspace is supposed to be located. The variable SCHEDWS contains the name of the SCHED workspace.

C.1.5 APL Provided Functions

The following functions are provided with the APL*PLUS system and provide an interface with the Macintosh Toolbox. Details of their operation is provided in [STSC, 1986].

BUTTON	Indicates whether or not the mouse button is depressed.
CLIPRECT	Sets a boundary for drawing.
CUTPICTURE	Returns a QuickDraw picture of the specified portion of the screen.
DELETEMENU	Removes the specified menu number from the menu bar.
DRAWLINE	Draws a line between two specified locations.
DRAWMENUBAR	Redraws the menu bar.
DRAWPICTURE	Draws a picture in the specified portion of the screen.
DRAWTEXT	Draws text at a specified location
ERASERECT	Fills the inside of the specified rectangle with the background pattern.
FILLRECT	Fills the inside of the specified rectangle with the specified pattern.
FRAMERECT	Draws the outline of the specified rectangle.
FRAMEROUNDRECT	Draws the outline of the specified rounded rectangle.
GETMOUSE	Returns the pixel position of the mouse on the screen.
GETPEN	Returns the pixel position of the pen.
HIDECURSOR	Makes the cursor invisible.
INITCURSOR	Sets the cursor to the standard mouse cursor and makes it visible.
INVERTRECT	Inverts all the pixels inside the specified rectangle.
INVERTROUNDRECT	Inverts all the pixels inside the specified rounded rectangle.
LINETO	Draws a line from the current pen location to the specified spot.
MOVETO	Moves the pen from the current pen location to the specified spot.
PAINTRECT	Fills the inside of the specified rectangle with the current pen pattern.
PENMODE	Sets the pen mode, which determines graphics overlap.
PENNORMAL	Sets the pen size, mode, and pattern to original values.
PENPAT	Changes the current pen pattern.
PENSIZE	Sets the size of the QuickDraw pen for drawing all lines and frames.
PICTFRAME	Returns the original frame size of a QuickDraw picture.
PTINRECT	Indicates whether a point is inside a specified rectangle.
SCROLLRECT	Moves the specified rectangle the specified number of pixels.
SETMENU	Puts a new menu on the menu bar.
SHOWCURSOR	Makes the cursor visible.

STANDMENUBAR	Returns the menu bar to the standard APL*PLUS menu configuration.
TEXTFACE	Specifies the style of subsequent text.
TEXTFONT	Specifies the font of subsequent text.
TEXTSIZE	Specifies the point size of subsequent text.
TEXTWIDTH	Measures the width of a specified character vector in pixels.

C.1.6 Data Manipulation Functions

The following functions all perform manipulation on character data:

Function Name	Arguments	Description
ADJOIN	A, B	Returns a matrix containing A and B adjoined to each other side by side. A and B can be character scalars, vectors, or matrices.
ADJOIN2	A, B	Same as ADJOIN, but A and B must be matrices, with A having at least as many rows as B. Works faster than ADJOIN.
APPEND	A, B	Returns a matrix containing A and B appended with A above B. A and B can be character scalars, vectors, or matrices.
BL	S	Takes a character vector S and removes any multiple occurrences of the space character from the vector.
CAPPEND	A, B	Same as APPEND, but centers the B matrix beneath the A matrix.
CHARMAT	B, V	Takes the character vector V and changes it into a matrix where the characters in the vector B are taken as delimiters signalling the start of a new line.

CLIST	A, B	Draws the character vector or matrix A at the coordinates indicated by B.
CLIST1	A, B	Same result as CLIST, but B is always a vector or a single row matrix. Works faster than CLIST.
CONVT	DATE	Attempts to interpret the character vector DATE as a date and time in the form {month day, year hour:minute}. If successful, the function will return the date as the number of minutes since January 1, 1900. If unsuccessful, the function will return the number -1.
CONVT2	DEF, DATE	Same as CONVT, but this function uses the date specified in DEF to provide default values, if needed, for the year, month, and day.
CVTDATE	DEF, T	Takes a numeric vector of numbers T, each element of which specifies a number of minutes since January 1, 1900, and converts this to a matrix, each row of which contains a character vector in the form Month Day, Year, corresponding to each element in T. Any entries in T which are equal to DEF are converted to the vector **NO INFO**.
CVTDATE1	DEF, T	Same as CVTDATE, but T can only be a single number, not a vector of numbers. Works faster than CVTDATE.
CVTNUM	DEF, NUMS	Converts a vector of numbers NUMS to a character matrix, where each row in the matrix contains a character vector of the corresponding number in NUMS. Any entries in NUMS which are equal to DEF are converted to the vector **NO INFO**.
CVTNUM1	DEF, NUMS	Same as CVTNUM, but NUMS must be a single number, not a vector of numbers. Works faster than CVTNUM.

CVTTIME	DEF, T	Inverse of the function CONVT. Takes a numeric vector of numbers T, each element of which specifies a number of minutes since January 1, 1900 0:00, and converts this to a matrix, each row of which contains a character vector in the form Month Day, Year Hour:Minute, corresponding to each element in T. Any entries in T which are equal to DEF are converted to the vector **NO INFO**.
CVTTIME1	DEF, T	Same as CVTTIME, but T can only be a single number, not a vector of numbers. Works faster than CVTTIME.
DLBS	A	Removes the first blank space, and everything following it, from the character vector A.
DLBS2	A	Removes all trailing blank spaces from the character vector A.
DLBS3	A	Removes all trailing blank columns from the character matrix A.
DLBS4	A	Removes all leading and trailing blank spaces from the character vector A.
ELIM	A, B	If A is a vector, eliminates the characters or numbers in the B position(s) of the vector A. If A is a matrix, then the rows indicated by B are eliminated.
FIND	A, B	A and B are character matrices. Returns a 0-1 vector containing a 1 for each row in B which has an exact match in one of the rows in A (trailing spaces are added as necessary). Contains a 0 of each row in B which does not have an exact match in A.

FIND3	W, T	W is a vector character string, T is a character matrix. Returns the row numbers in T in which the character string W appears.
UPPERCASE	A	Changes all lowercase letters in the character vector A to uppercase letters.

C.1.7 The Input Interface Functions

The dyadic function GETINP handles all typed input into MFIVE (with the exception of the text entered via the function Z, as described in Section C.1.8). The function GETINP is called from the function GETINPUT. GETINPUT is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of GETINP. GETINP has two arguments, B and A. The numeric vector A is a four element vector which describes the location on the screen where the input characters are to be displayed. The numeric vector B describes the type of input being requested. The function GETINP returns an explicit value, which corresponds to the input, as well as a second value, assigned to the variable OPT. If OPT is set equal to 1, it indicates that the input received was of the type stipulated in the variable B. If OPT is set equal to 2, it indicates that the return key was hit without any characters being entered. If OPT is set equal to 3, it indicates that the word "delete" was entered. Finally, if OPT is set equal to 6, it indicates that the word "help" was entered.

The first element in the vector B describes the type of input requested, and subsequent elements in the vector, if present, describe additional restrictions on the input. The following are valid options for the variable B:

B[1] = 1 This requests input of a character string. The output is the character vector which was input. If B[2] = 0 (or is nonexistent), then any characters may be entered. If B[2] = 1, then any characters in the character vector VAL are not allowed as input. The character vector VAL is assigned by the program calling GETINP. If B[3] = 0 (or is nonexistent), then there is no limit on the length of the input. If B[3] is a positive integer, then that integer signifies the maximum number of characters allowed in the input.

B[1] = 2 This requests input of a date, in the form Month, Day Year. The output is the date, expressed as the number of minutes from January 1, 1900. The month name may be entered numerically or spelled out. B[2], if present, is a default date (in minutes, from January 1, 1900) passed on to the function CONVT2 (see Section C.1.6). B[3], if present and greater than zero, specifies a minimum date. The input date must be at or after this date. B[4], if present, specifies a maximum date. The input date must be before or equal to this date.

B[1] = 3 This requests input of a date and time, in the form Month, Day Year Hour:Minute. The output is the date and time, expressed as the number of minutes from January 1, 1900 0:00. The month name may be entered numerically or spelled out. B[2], if present, is a default date and time (in minutes, from January 1, 1900 0:00) passed on to the function CONVT2 (see Section C.1.6). B[3], if present and greater than zero, specifies a minimum date and time. The input date and time must be at or after this date. B[4], if present, specifies a maximum date and time. The input date and time must be before or equal to this date.

B[1] = 4 This requests input of a numerical value or values. The output is a numeric vector containing the numeric value(s) entered. If B[2] is nonexistent or equal to zero, then there is no limit on the number of values which can be entered. If B[2] is a positive integer, than that integer specifies the number of entries which are to be entered. If B[3] is nonexistent or equal to zero, then the entries do not have to be integer. If B[3] is nonzero, then all the entries must be integers. B[4], if it exists, specifies a minimum. All entries must be greater than or equal to this value. B[5], if it exists, specifies a maximum. All entries must be less than or equal to this value. If it is desired to specify a maximum value, but no minimum value, then this is signaled by making B[4] greater than B[5]. B[4] will then be ignored.

B[1] = 5 This requests input in the form YES (or Y) or NO (or N). The output is 1 if YES is input, and 0 if no is input.

B[1] = 6 This requests input off of lists corresponding to the information forms. For example, when entering the name of a crewmember, it is checked to see if there is any type of match between the text typed in and names and nicknames on the crewmember list (in the variables CREWLIST and CREWN). The match need not be complete, and differences between uppercase and lower case characters are ignored. The output is the inventory number of the entry typed in. If an entry is typed in which is ambiguous, (e.g., if "Dan" is typed in and there are two Dans on the list), then a window will be posted showing all the valid choices and asking the user to select the correct one. By typing in "help", the facility in GETINP will provide help assistance by posting a list of valid entries and asking the user to select from the list with the mouse. B[2] should contain the type of information form which the entry is being selected from. It should be equal to 1 for the crew form, 2 for the job form, 3 for the flight plan form, 4 for the tool form, and 5 for the stowage form. B[3], if present, specifies the maximum number of entries which can be entered. B[4], if present and equal to 1, allows for the entering of a new entry (i.e., adding a new crewmember). Otherwise B[4] should be set to 0. Any components of B after the fourth element, if present, specify the inventory numbers of elements of the list (e.g. CREWLIST) which are not allowed to be selected.

B[1] = 7 This requests inputs off lists which are not information forms. The character vector VAL is a pointer to the list that is assigned by the calling program and should contain the prefix of the name of the list from which input is requested. For example, if VAL is given the value "CREWRANK", then the list is found in the variable CREWRANKLIST, nicknames for items on the list (if any) are found in the variable CREWRANKN, and the inventory numbers of the objects on the list are found in the variable CREWRANKNN. B[2], if present, specifies the maximum number of entries which can be entered. B[3], if present and equal to 1, allows for the entering of a new entry (i.e., adding a new crewmember). Otherwise B[3] should be set to 0. Any components of B after the third element, if present, specify the inventory numbers of elements of the list (e.g. CREWRANKLIST) which are not allowed to be selected.

B[1] = 8 This requests input in the form Day/Hours:Minutes. Output is in units of minutes from time zero. B[2], if present, is used to specify default values for the day and hours. B[3], if present, is used to specify a minimum entry, and B[4], if present, is used to specify a maximum entry. This option (B[1] = 8) is not used in the PRIME workspace. It requires the presence of the function CONVT3, which is found in the SCHED workspace.

The dyadic function REPLYWINDOW is used to display a message and accept a typed response. A window is created at the top of the screen, where the message is displayed, and input is entered. After accepting input, the window is removed, and the screen restored to its original condition. REPLYWINDOW utilizes the function GETINPUT (and therefore GETINP also). REPLYWINDOW has two arguments, the variables B and A. The numeric variable B has the same format as the variable B for GETINP, and thus characterizes the type of input desired. The variable A is a character vector which contains the message to be displayed on the screen. As with GETINP, REPLYWINDOW returns the input value as an explicit result, and the variable OPT implicitly.

The monadic function ERRORWINDOW is used to display a message and await for the user to click on the word "OK" in response. A window is created at the top of the screen, where the message is displayed. After the user clicks "OK", the window is removed, and the screen restored to its original condition. ERRORWINDOW has a single argument, the variable A, which is a character vector containing the message to be displayed. ERRORWINDOW does not return an explicit result.

The monadic function HELPWINDOW is also used to display a message and await for the user to click on the word "OK" in response. A window is created at the top of the screen, where the message is displayed. After the user clicks "OK", the window is removed, and the screen restored to its original condition. HELPWINDOW has a single argument, the variable A, which is a character vector containing the message to be displayed. HELPWINDOW does not return an explicit result. HELPWINDOW accepts longer (multi-line) messages than can ERRORWINDOW. The vector A should contain a return character at each intended line break.

The monadic function YESNOWINDOW is used to display a message and await for the user to click on either the word YES or NO in response. A window is created at the top of the screen, where the message is displayed. After the user clicks YES or NO, the window is removed, and the screen restored to its original condition. YESNOWINDOW has a single argument, the variable A, which is a character vector containing the message to be displayed. YESNOWINDOW returns an explicit result of 0 if NO is clicked, or 1 if YES is clicked.

C.1.8 Other User Interface Functions

The monadic function CH allows the user to select off of a list like the one that is displayed when one tries to assign crewmembers or jobs to a flight plan (see Figure 6.12). The function CH is called from the function CHOICE. CHOICE is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of CH. CH has a single explicit input arguments, D. Other variables are, however, implicitly passed on to CHOICE (and CH) from the calling program. The character matrix LIST should contain the list that is to be selected from. The numeric variable MAX should contain the maximum number of entries that are to be allowed to be selected from LIST. If MAX is specified as 0, then there is no limit placed on the number of entries (except, of course, by the size of the list). The character vector MSG should contain the name of the window that is being opened, and the character MSG2 should contain the message to the user that is displayed in the right hand side of the window. The numeric vector D should contain the row numbers of the items on the list which have already been selected (and should therefore show as inverted). The function CH returns an explicit result which is the row numbers of the items on the list which have been selected. The function CH uses the niladic function IR2, which directs what is done if the user clicks the mouse on the list inside the window presented by CH. IR2 does not return an explicit result.

The monadic function EDIT allows the user to add, delete, and modify nickname lists (see Figure 6.10). The function EDIT is called from the function ED. ED is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of EDIT. EDIT has a single explicit input argument, Q. The first word in the Q vector is the name of the variable in which the list is stored. This variable is implicitly passed from the calling program to EDIT. All text in Q after the first word constitutes the name of the window, which is displayed on the window's top line. The function EDIT produces no explicit value, but instead modifies the list which is implicitly passed back to the calling program. The function EDIT uses the niladic function ED2, which directs what is done if the user clicks the mouse on the list inside the window, in order to modify the nicknames. ED2 returns an explicit value of 0 or 1. If the value is 0 it indicates to EDIT that the revisions are extensive enough so that the entire inside of the window must be redrawn.

The dyadic function IR is used by CH and EDIT to oversee the scrolling of sections of the window which are shown inverted. IR has two arguments, A and B, which specify which portions of the nickname list are currently showing on the screen. It does not return an explicit result.

The dyadic function EDITA allows the user to modify numeric entries associated with a list like the one that is displayed when one tries to modify crewmember or job performance times (see Figure 6.6). The function EDITA is called from the function EDA. EDA is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of EDITA. EDITA has two explicit input arguments, P and Q. Other variables are, however, implicitly passed on to EDA (and hence EDITA) from the calling program. The character vector MSG2 contains the message displayed at the top of the window, directly beneath the line containing the name of the window. The variable HELPMSG contains the message to be displayed if the user asks for help. The names of the other variables which are passed to EDITA are contained in the character vector Q. The first word in the Q vector is the name of the variable in which the list is stored. The second word in the Q vector is the name of the variable in which is stored the message which is to appear when the user clicks on an item in the list. Finally, all text in Q after the second word constitutes the name of the window, which is displayed on the window's top line. The numeric vector P contains the values associated with each item on the list which are being modified. The function EDITA returns an explicit result which contains the new values associated with the items on the list. The function EDITA uses the niladic function ED2A, which directs what is done if the user clicks the mouse on the list inside the window, in order to modify a numeric values. ED2A returns an explicit value of 0 or 1. If the value is 0 it indicates to EDITA that the revisions are extensive enough so that the entire inside of the window must be redrawn.

The function EDITB, and its shielding function EDB, are very nearly identical with EDITA and EDA. However, in addition the the other variables passed from the calling program, the variable character vector MSG3 is also implicitly passed. This vector contains a message that is displayed if the user tries to click on the text within the window. These functions are used by the tool searcher to display a list and get a YES or NO response from the user (Figure 6.50). It returns an explicit result of 1 for yes, 0 for no.

The monadic function Z allows the display and modification of text windows like that used for crewmember background information (see Figure 6.5). The function Z is called from the function Z1. Z1 is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of Z. Z has a single explicit input argument, C. C contains a character matrix with the initial form of the text that is to be displayed in the window. The function Z returns an explicit result, which is a character matrix containing the revised form of the text.

The niladic function TOOLCONT allows the display and modification of Default Loaction and Status window on the Tool Information Form (see Figure 6.15). The function TOOLCONT is called from the function TOOLCONT1, which is in turn called from the function TOOLCONT2. TOOLCONT1 and TOOLCONT2 are "shielding functions": their only purpose is to allow the declaration of more local variables than could comfortably be included in the header of TOOLCONT. When the user clicks the mouse inside the TOOLCONT window, revision of information appearing there is accomplished by the function TOOLW. TOOLW returns an explicit value of 0 or 1. If the value is 0 it indicates to TOOLCONT that the revisions are extensive enough so that the entire inside of the window must be redrawn.

The function DRAWMSG is used by all the window producing functions in this section to write the name of the window in the bar at the window's top. DRAWMSG prevents the window name from running outside the window (by truncation) if it is longer than can fit inside the bar. DRAWMSG has two arguments, A and B. The character vector A contains the name of the window that is to be writen. B is a 2 element numeric vector. B[1] contains the number of pixels which will be needed to (fully) draw character vector A, and B[2] contains the number of pixels actually available to draw the name.

C.1.9 Data Transfer to the Scheduler

The function MANSCHED is activated when the schedule button is clicked on a Flight Plan Information Form. MANSCHED prepares data for the scheduler, checks for infeasibilities, and then transfers control to the SCHED workspace. MANSCHED has a single argument, OBJ, which is the object number of the flight plan being sent to the scheduler.

The function WS returns the name of the Macintosh folder in which the PRIME folder (and presumably the SCHED folder) are located. The function SCHED loads the SCHED workspace.

C.1.10 The Tool Searcher Functions

The functions SEARCHCTL, SEARCHCVTNUM, SEARCHDISPLAY, SEARCHHISTRANK, SEARCHINFER, SEARCHJOBRANK, SEARCHPROXRANK, and SEARCHSIZERANK are all used by the tool searcher function TOOLSEARCH which is activated when the user clicks the mouse on the SEARCH button on the Tool Information Form. TOOLSEARCH has a single argument, OBJ, which is the object number of the tool being searched for. All the tool searching functions are extensively documented in [Kranzler, 1986].

C.1.11 File Manipulation Functions

The function FILEOPEN is a monadic function which is used to get from the user (via a dialog box) the name of a file (containing a database) which is to be loaded into MFIVE. The function has a single argument, P, which is a character vector containing the prefix (ther first letters) which the file name must possess. The function returns an explicit result, which is the name of the file to be opened.

The function DATALOAD is a monadic function which load the database into MFIVE. The function has a single argument, W0, which is the name of the file to be loaded. The fuction has no explicit result.

The function FILESAVE is a dyadic function which is used to get from the user (via a dialog box) the name of a file a MFIVE database is to be saved. The function has two arguments, A and B. The variable B contains a character vector containing the prefix (the first letters) which the file name must possess. The variable A contains a message displayed to the user in the dialog box. The function returns an explicit result, which is the name of the file to be opened.

The function DATASAVE is a dyadic function which saves an MFIVE database. The function has two arguments, W7 and KK. W7 is a character vector which contains the name of the file in which the database is to be saved. KK is a character matrix containing the names of the variables to be saved in the file. The function returns an explicit result, which is the total number of variables saved (rows in the KK matrix) plus 1.

The niladic function PRIMEDATA produces an explicit result which is a character matrix containing the names of all the variables which need to be saved when saving a database. This matrix is then passed to the function DATASAVE.

The dyadic function FILEHCREATE is used by the function MANSCHED to create the files necessary to transfer data from the PRIME workspace to the SCHED workspace. FILEHCREATE is has two arguments, A and B. The character vector A contains the name of the file to be created. The number B contains the number of file to be opened. FILEHCREATE does not return an explicit result.

C.1.12 Miscellaneous Functions

The function TEXTNORMAL returns the textfont, textsize, and textface to a preset arrangement.

The function T resets text parameters, the cursor, pen, and menu bar to a preset arrangement. It also unties any tied files.

The function CLEAR draws the word MFIVE in the bar at the top of the screen, and defines the size of the screen used by MFIVE.

The function DRAWMFIVEICON draws the MFIVE icon and the SSL logo on the screen.

The function HCL is used to produce screen or printer listing of APL functions. HCL has a single argument, INS. INS is a character vector which contains the names of the functions to be listed, separated by spaces. By specifying INS to be the character string "ALL", all of the functions in the workspace will be listed, followed by a listing of all variable names.

C.1.13 Function Listings for the PRIME Workspace

```
      )FNS
ADJOIN           ADJOIN2          APPEND           BL               BUTTON
CAPPEND          CH               CHARMAT          CHOICE           CLEAR
CLIPRECT         CLIST            CLIST1           CONVT            CONVT2
CREWASGET        CREWBDGET        CREWHTGET        CREWINFOGET      CREWMSGET
CREWPERFGET      CREWRANKGET      CUTPICTURE       CVTDATE          CVTDATE1
CVTNUM           CVTNUM1          CVTTIME          CVTTIME1         DATALOAD
DATASAVE         DELETEMENU       DLBS             DLBS2            DLBS3
DLBS4            DRAWLINE         DRAWMENUBAR      DRAWMFIVEICON    DRAWMSG
DRAWPICTURE      DRAWTEXT         ED               ED2              ED2A
EDA              EDB              EDIT             EDITA            EDITB
ELIM             ERASERECT        ERRORWINDOW      FILEHCREATE      FILEOPEN
FILESAVE         FILLRECT         FIND             FIND3            FORMGET
FRAMERECT        FRAMEROUNDRECT   GETINP           GETINPUT         GETMOUSE
GETNAME          GETPEN           HCL              HELPWINDOW       HIDECURSOR
ICON             INITCURSOR       INVERTRECT       INVERTROUNDRECT  IR
IR2              JOBAUDIOGET      JOBDEFGET        JOBINFOGET       JOBMEMORYGET
JOBMULTIPLEXGET  JOBPERFGET       JOBPOWERGET      JOBSIZEGET       JOBTOOLGET
JOBTRANSGET      LINETO           MANASGET         MANCREWGET       MANJOBGET
MANMEMORYGET     MANMULTIPLEXGET  MANPOWERGET      MANSCHED         MANTRANSGET
MOVETO           PAINTRECT        PENMODE          PENNORMAL        PENPAT
PENSIZE          PICTFRAME        PRIMEDATA        PTINRECT         REPLYWINDOW
REVISENAME       REVISENICKNAME   S                SCHED            SCROLLRECT
SEARCHCTL        SEARCHCVTNUM     SEARCHDISPLAY    SEARCHHISTRANK   SEARCHINFER
SEARCHJOBRANK    SEARCHPROXRANK   SEARCHSIZERANK   SETMENU          SHOWCURSOR
STANDMENUBAR     STOWMODGET       STOWTHETAGET     STOWTOOLGET      STOWVOLGET
STOWXGET         T                TEXTFACE         TEXTFONT         TEXTNORMAL
TEXTSIZE         TEXTWIDTH        TOOLCONT         TOOLCONT1        TOOLCONT2
TOOLINFOGET      TOOLSEARCH       TOOLSTOWGET      TOOLUSEGET       TOOLVOLGET
TOOLW            UPPERCASE        WS               XCREWFORMFILL    XJOBFORMFILL
XMANFORMFILL     XSTOWFORMFILL    XTOOLFORMFILL    YESNOWINDOW      Z
Z1
```

```
     ∇R←A ADJOIN B;D;D1;D2
[1]   D1←NTWO↑ONE,ρA⋄A←D1ρA⋄D2←NTWO↑ONE,ρB⋄B←D2ρB⋄D←ONE↑D1⌈D2
[2]   R←((D,D1[ONE])↑A),(D,D2[TWO])↑B
     ∇
```

```
     ∇R←A ADJOIN2 B
[1]   R←A,((ρA)[ONE],(ρB)[TWO])↑B
     ∇
```

```
     ∇R←A APPEND B;D;D1;D2
[1]   D1←NTWO↑ONE,ρA⋄D2←NTWO↑ONE,ρB⋄A←D1ρA⋄B←D2ρB⋄D←NONE↑D1⌈D2
[2]   R←((D1[ONE],D)↑A)⍪(D2[ONE],D)↑B
     ∇
```

```
     ∇R←BL S;C
[1]   C←S≠' '⋄R←(C∨ONE⌽C)/S
```

⍒BUTTON⍒

```
     ∇R←A CAPPEND B;D;E;F;G
[1]   E←NTWO↑ONE,⍴A◇A←E⍴A◇F←NTWO↑ONE,⍴B◇B←F⍴B◇→(ZERO≠(⍴B)[ONE])/B1◇B←B⍪' '◇F←⍴B
[2]   B1:D←NONE↑E⌈F◇G←PFIVE×NONE↑F-E
[3]   R←((ZERO⌊-⌊G)⌽(E[ONE],D)↑A)⍪(ZERO⌊⌈G)⌽(F[ONE],D)↑B
     ∇


     ∇S←CH D;AMAX;S1;S2;S3;S4;OUT;LC;A;ULC;B;RECT;MS;CR;TR;C;UN;UN2;C1;P;RS
[1]   AMAX←(⍴LIST)[ONE]◇MAX←MAX+(MAX=ZERO)×AMAX◇N←18◇OUT←18 0⍴IZ◇K←ZERO
[2]   B31:OUT←OUT ADJOIN2 (DLBS3 LIST[K+⍳N⌊AMAX-K;]),AV125◇K←K+N◇→(AMAX>K)/B31
[3]   FL←NONE,(OUT[ONE;]=AV125)/⍳(⍴OUT)[TWO]◇MSG←' ',MSG,' '◇S←(D>ZERO)/D
[4]   OUT←0 ¯1↓OUT◇TEXTSIZE 12◇TEXTFONT ZERO◇TEXTFACE ZERO◇MS←TEXTWIDTH MSG
[5]   SIZE←11 6×1 2+⍴OUT◇ULC←60 100◇C1←L1,L2,L3,L4,L5,L6,L7,L8,L2,L10◇P←⍳TEN
[6]   SIZE[TWO]←SIZE[TWO]⌊246◇LC←ZERO◇RC←⌊(¯9+SIZE[TWO])÷SIX◇RS←(D<ZERO)/D
[7]   B4:S1←ULC[ONE]◇S2←ULC[TWO]◇S3←S1+SIZE[ONE]◇S4←S2+SIZE[TWO]
[8]   RECT←S1,S2,S3,S4◇B←¯18 0 18 168+RECT◇UN←TWO↑B◇UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
[9]   K←⎕EX 'UN2'◇UN2←⎕GETBITS UN◇ERASERECT B◇TR←¯18 0 1 168+S1,S2,S1,S4
[10]  CR←¯14 10 ¯3 22+S1,S2,S1,S2◇G3←¯1 0 18 18+S3,S2,S3,S2
[11]  G3 DRAWPICTURE LEFTARROW◇G4←¯1 ¯18 18 0+S3,S4,S3,S4
[12]  G4 DRAWPICTURE RIGHTARROW◇K←(S1-15)+TWO×⍳SIX
[13]  K←⍉4 6⍴(K-ONE),(SIX⍴TWO+S2),K,SIX⍴S4+165◇RECT←RECT⍪5 4⍴CR,TR,G3,G4
[14]  FRAMERECT K⍪B⍪RECT◇G1←⌊PFIVE×168+S4+S2-MS⌊S4+123-S2◇TEXTSIZE 12
[15]  K←(S1-17),G1,(S1-ONE),168+S4+S2-G1
[16]  ERASERECT K⍪3 4⍴(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
[17]  MOVETO (S1-THREE),G1◇TEXTFONT ZERO◇MSG DRAWMSG MS,MS⌊S4+123-S2◇TEXTNORMAL
[18]  GRAYPAT FILLRECT 0 18 17 ¯18+S3,S2,S3,S4◇TEXTFACE ONE◇K←ONE◇C←MSG2
[19]  B20:MOVETO (NINE+S1+K),S4+TEN◇A←23-(⌽23↑C)⍳' '◇DRAWTEXT A↑C◇C←(ONE+A)↓C
[20]  K←K+TEN◇→(ZERO≠⍴C)/B20◇C←(15+S1+K),S4+20◇MOVETO C-0 10
[21]  DRAWTEXT 'MAX SELECTABLE = ',FOUR↑⍕MAX⌊AMAX-⍴RS◇C←C+10 0◇MOVETO C-0 10
[22]  DRAWTEXT 'NUMBER SELECTED = ',FOUR↑(⍕⍴S)
[23]  A←(C+10 10),(C+25 110),15 15◇K←¯2 ¯2 2 2 2 2◇E←A,A+K
[24]  MOVETO 11 15+A[1 2]◇DRAWTEXT 'SELECT ALL'◇RECT←RECT⍪1 4⍴A
[25]  A←A+23 0 23 0 0 0◇E←E,A,A+K
[26]  MOVETO 11 20+A[1 2]◇DRAWTEXT 'CLEAR ALL'◇RECT←RECT⍪1 4⍴A
[27]  A←A+23 ¯15 23 ¯58 0 0◇E←E,A,A+K
[28]  MOVETO 11 15+A[1 2]◇DRAWTEXT 'DONE'◇RECT←RECT⍪1 4⍴A
[29]  A←A+0 75 0 75 0 0◇E←E,A,A+K
[30]  MOVETO 11 9+A[1 2]◇DRAWTEXT 'CANCEL'◇RECT←RECT⍪1 4⍴A
[31]  FRAMEROUNDRECT 8 6⍴E◇TEXTFACE ZERO
[32]  B14:MOVETO 10 8+S1,S2◇DRAWTEXT (N,RC)↑(ZERO,LC)↓OUT◇ZERO IR RC
[33]  B18:G5←(RC×¯36+S4-S2)÷TWO×(⍴OUT)[TWO]
[34]  G2←S2+N+((LC+PFIVE×RC)÷(⍴OUT)[TWO])×S4-S2+36
[35]  G5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◇ERASERECT G5◇FRAMERECT G5
[36]  RECT[SIX;]←G5
[37]  B2:M←⎕GETKEY◇→(THREE≠⍴M)/B2◇→(TWO=M[ONE])/B2◇M←M[2 3]
[38]  L←ONE↑(M PTINRECT RECT)/P◇→(ZERO=L)/B2◇→C1[L]
[39]  B3:⎕SOUND BEEP◇→B2
[40]  L1:→BUTTON/L1◇IR2◇→B2
[41]  L2:→BUTTON/L2◇→(~GETMOUSE PTINRECT RECT[L;])/B2◇UN2 ⎕PUTBITS UN◇→ZERO
[42]  L3:M1←B◇PENPAT GRAYPAT◇PENMODE TEN
[43]  B7:→(~BUTTON)/B5◇FRAMERECT M1◇M1←B+FOUR⍴(0 0⌈GETMOUSE)-M◇FRAMERECT M1◇→B7
[44]  B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
[45]  B30:FRAMERECT M1◇UN2 ⎕PUTBITS UN◇PENPAT BLACKPAT◇PENMODE EIGHT◇→B4
[46]  L4:→(LC≤ZERO)/B3◇HS←(⌽(LC-ONE)↑OUT[ONE;])⍳AV125◇G3←LC⌊HS
[47]  SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO◇LC←LC-G3
[48]  MOVETO 10 8+S1,S2◇DRAWTEXT (N,G3⌊RC)↑(ZERO,LC)↓OUT
```

```
[49]  ZERO IR RC⌊G3◊GRAYPAT FILLRECT G5◊→B18
[50]  L5:G3←(⍴OUT)[TWO]-RC◊→(LC≥G3)/B3◊HS←((RC+LC+ONE)↓OUT[ONE;])⍳AV125◊G3←HS⌊G3
[51]  SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(¯6×G3),ZERO◊LC←LC+G3
[52]  MOVETO (S1+TEN),S2+EIGHT+SIX×RC-G3◊DRAWTEXT (N,-G3)↑(N,RC)↑(ZERO,LC)↓OUT
[53]  (ZERO⌈RC-G3) IR RC◊GRAYPAT FILLRECT G5◊→B18
[54]  L6:M1←G5◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[55]  B23:→(~BUTTON)/B24◊FRAMERECT M1
[56]  K←(18+S2-G5[2])⌈(S4-18+G5[4])⌊1↓GETMOUSE-M◊M1←G5+4⍴0,K◊FRAMERECT M1◊→B23
[57]  B24:FRAMERECT M1◊GRAYPAT FILLRECT G5◊PENPAT BLACKPAT◊PENMODE EIGHT
[58]  LC←LC+⌊PFIVE+K×(⍴OUT)[TWO]+S4-S2+36◊ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◊→B14
[59]  L7:→(MAX<AMAX-⍴RS)/B32◊K←S◊S←(~(⍳AMAX)∊S,-RS)/⍳AMAX◊ZERO IR RC◊S←K,S◊→B33
[60]  B32:K←'YOU CANNOT SELECT MORE THAN ',(⍕MAX),' ENTR',(3+¯4×ONE=MAX)↑'IESY'
[61]  ERRORWINDOW K◊→B2
[62]  L8:ZERO IR RC◊S←IZ
[63]  B33:ERASERECT (C+¯9 116),C+0 145◊MOVETO C+0 116◊TEXTFACE ONE
[64]  DRAWTEXT FOUR↑⍕⍴S◊TEXTFACE ZERO◊→B2
[65]  L10:S←(D>ZERO)/D◊→L2
    ∇


    ∇Z←B CHARMAT V;A
[1]   V←V,(,B)[ONE]◊A←V∊B
[2]   Z←(⍴A)⍴(,A←A∘.≥⍳⌈/0,A←(A≠0)/A←A-1+0,NONE↓A←A/⍳⍴A)\(~A)/V
    ∇


    ∇S←CHOICE D;K;G1;G2;G3;G4;G5;L;M;N;M1;RC;E;FL;HS;SIZE
[1]   S←CH D
    ∇


    ∇CLEAR;K;KK
[1]   K←ONE◊HIDECURSOR◊KK←503132+524288×⌊PFIVE+⎕WSSIZE÷524288
[2]   CLIPRECT 0 0 299 507
[3]   B1:MFIVEHEADER[K;] ⎕POKE KK+K×64◊K←K+ONE◊→(K≤NINE)/B1◊SHOWCURSOR
    ∇


  ∇CLIPRECT∇


    ∇A CLIST B;C;K
[1]   →(ZERO=×/⍴A)/ZERO◊ERASERECT B◊TEXTFACE ONE◊A←(NTWO↑ONE,⍴A)⍴A
[2]   C←(EIGHT+⌈PFIVE×B[THREE]+B[ONE])-FIVE×ONE↑⍴A◊K←ZERO
[3]   MOVETO C,B[TWO]+FIVE◊DRAWTEXT A◊TEXTFACE ZERO
    ∇


    ∇A CLIST1 B
[1]   ERASERECT B◊TEXTFACE ONE
[2]   MOVETO (THREE+⌈PFIVE×B[THREE]+B[ONE]),B[TWO]+FIVE◊DRAWTEXT A◊TEXTFACE ZERO
    ∇


    ∇TET←CONVT DATE;MO;VI;FI;YR;MIN;DAY;HR
[1]   TET←NONE◊DATE←DLBS4 ,DATE◊DATE[(DATE∊'/,:')/⍳⍴DATE]←' '
[2]   VI←,⎕VI DATE◊→(FIVE≠⍴VI)/ZERO◊FI←⎕FI DATE◊→(~VI[ONE])/L1
[3]   MO←ONE↑FI◊→(∨/(MO≤ZERO),(12<MO),MO≠⌊MO)/ZERO◊→L2
```

```
[4]   L1:MO←,(∧/MONTHS=12 3ρDATE[1 2 3])ιONE◊→(13=MO)/ZERO
[5]   L2:YR←FI[THREE]◊HR←FI[FOUR]◊DAY←FI[TWO]-ONE◊YR←YR-1900×YR≥1900
[6]   →(∨/(HR,DAY,YR)≠⌊HR,DAY,YR)/ZERO
[7]   →(∨/(HR,DAY)>23,NONE+MOS2[MO]+(MO=TWO)∧ZERO=FOUR|YR)/ZERO◊MIN←FI[FIVE]
[8]   →(∨/(ZERO∊ONE↓VI),(YR<ZERO),(ZERO>HR,MIN,DAY),MIN≥60)/ZERO
[9]   TET←MIN+60×HR+24×DAY+(ZERO,,MOS)[MO]+((MO>TWO)∧ZERO=FOUR|YR)+⌈365.25×YR
    ∇


    ∇TET←DEF CONVT2 DATE;VI;FI;YEAR;MO;DAY;HR;MIN;F
[1]   →(DEF≥ZERO)/L2
[2]   L1:TET←CONVT DATE◊→ZERO
[3]   L2:TET←NONE◊DATE←DLBS4 ,DATE◊DATE[(DATE∊'/,:')/ιρDATE]←' '◊VI←,⎕VI DATE
[4]   →(FIVE=ρVI)/L1◊→((TWO>ρVI)∨FIVE<ρVI)/ZERO◊→(ZERO∊(FOUR=ρVI)↓VI)/ZERO
[5]   DEF←,⍉0 365.25 24 60⊤DEF◊YEAR←1↑DEF◊FI←,⎕FI DATE◊→(FOUR≠ρVI)/L4
[6]   →(~VI[ONE])/L3◊MO←FI[ONE]◊→(∨/(MO≤ZERO),(12<MO),MO≠⌊MO)/ZERO◊→L5
[7]   L3:→(13=MO←(∧/MONTHS=12 3ρDATE[1 2 3])ιONE)/ZERO◊→L5
[8]   L4:MO←ONE++/(12ρONE+⌊DEF[TWO])>F←,¯12 0↓(12×ZERO=FOUR|YEAR)∘MOS
[9]   L5:→(TWO=ρVI)ρL6◊DAY←FI[ONE+FOUR=ρVI]-ONE
[10]  →((DAY≠⌊DAY)∨(DAY≥MOS2[MO]+(MO=TWO)∧ZERO=FOUR|YEAR)∨ZERO>DAY)/ZERO◊→L7
[11]  L6:DAY←⌊DEF[TWO]-(ZERO,F)[MO]
[12]  L7:MIN←NONE↑FI
[13]  HR←ONE↑NTWO↑FI◊→(∨/(ZERO>HR,MIN),(HR>23),(MIN≥60),HR≠⌊HR)/ZERO
[14]  TET←MIN+60×HR+24×DAY+(0,,MOS)[MO]+((MO>TWO)∧ZERO=FOUR|YEAR)+⌈365.25×YEAR
    ∇


    ∇A CREWASGET OBJ;DEF;B
[1]   DEF←(CREWAS[A;OBJ]≠CREWASDEF[A])×CREWAS[A;0 J]-CREWAS[THREE-A;OBJ]
[2]   DEF←CREWAS[THREE-A;OBJ]+DEF
[3]   DEF←(THREE,DEF,((A=1)/0),(CREWAS  A;OBJ]≠CREWASDEF[3-A])ρCREWAS[3-A;OBJ])
[4]   B4:B←DEF GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[5]   CREWAS[A;OBJ]←B◊→ZERO
[6]   B1:(CREWASDEF[A] CVTTIME CREWAS[A;OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[7]   B2:CREWAS[A;OBJ]←CREWASDEF[A]◊→B1
[8]   B3:HELPWINDOW XCREWASHELP◊→B4
    ∇


    ∇CREWBDGET OJB;A
[1]   B4:A←(TWO,CREWBD[OBJ]) GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]   CREWBD[OBJ]←A◊→ZERO
[3]   B1:(CREWBDDEF CVTDATE CREWBD[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:CREWBD[OBJ]←CREWBDDEF◊→B1
[5]   B3:HELPWINDOW XCREWBDHELP◊→B4
    ∇


    ∇CREWHTGET OBJ;A
[1]   B4:A←4 1 0 137 214 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]   CREWHT[OBJ]←A◊→ZERO
[3]   B1:(CREWHTDEF CVTNUM CREWHT[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:CREWHT[OBJ]←CREWHTDEF◊→B1
[5]   B3:HELPWINDOW XCREWHTHELP◊→B4
    ∇


    ∇CREWINFOGET OBJ;B;MSG;C
[1]   B←'CREW',(⍕CREWNN[OBJ]),'INFO'◊→(TWO≠⎕NC B)/B1◊C←⍎B
```

```
[2]   B3:MSG←(DLBS2 ,CREWLIST[OBJ+ONE;]),' Background Information'
[3]   C←((TEN,⌈SIX+1.25×ρMSG)⌈ρC)↑C
[4]   C←Z1 C◊C←⍉DLBS3 ⍉DLBS3 C◊→(ZERO=×/ρC)/B2◊⍎B,'←C'◊→ZERO
[5]   B1:C←0 0ρ''◊→B3
[6]   B2:C←⎕EX B
     ∇


     ∇CREWMSGET OBJ;A
[1]   B4:A←4 1 0 45 114 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]   CREWMS[OBJ]←A◊→ZERO
[3]   B1:(CREWMSDEF CVTNUM CREWMS[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:CREWMS[OBJ]←CREWMSDEF◊→B1
[5]   B3:HELPWINDOW XCREWMSHELP◊→B4
     ∇


     ∇CREWPERFGET OBJ;G1;G2;G3;G4;C;D;J;E;H;VAL;R;K;F;G5;A;B;MSG2;M;HELPMSG
[1]   F←CREWNN[OBJ]◊G1←PERFDATA1=F◊C←G1/PERFDATA2◊G2←PERFDATA1<ZERO
[2]   J←G2/PERFDATA1◊J←-J[G1/+\G2]◊G3←JOBNNιJ◊HELPMSG←XCREWPERFHELP
[3]   H←JOBLIST[ONE+G3;]⍪JOBLIST ELIM ONE,ONE+G3◊E←G2/PERFDATA1◊G5←(~E∊-J)/E
[4]   M←PERFDATA2[PERFDATA1ιG5]◊M[(M=JOBTIMEDEF)/ιρM]←NTWO◊C←C,M◊D←ρC
[5]   G4←⍕(D,ONE)ρC◊G4[(C<ZERO)/ιD;]←' '◊G4←G4,(D,ONE)ρ((ρG3)ρ' '),(D-ρG3)ρ'*'
[6]   H←H,'/',G4◊VAL←'ENTER PERFORMANCE TIME:  '
[7]   MSG2←'Performance Times In Minutes'
[8]   R←C EDA 'H VAL ',BL CREWLIST[IZρONE+OBJ;],' Performance Data'
[9]   PERFDATA2[G1/ιρPERFDATA2]←(ρJ)↑R◊R←(ρJ)↓R◊A←(R≠(ρJ)↓C)/R◊G5←(R≠(ρJ)↓C)/G5
[10]  →(ZERO=ρA)/ZERO◊K←ONE
[11]  B1:B←PERFDATA1ιG5[K]◊PERFDATA1←(B↑PERFDATA1),F,B↓PERFDATA1
[12]  PERFDATA2←(B↑PERFDATA2),A[K],B↓PERFDATA2◊K←K+ONE◊→((ρA)≥K)/B1
     ∇


     ∇CREWRANKGET OBJ;A;VAL
[1]   B4:VAL←'CREWRANK'◊A←7 1 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3)/B1,B2
[2]   CREWRANK[OBJ]←A◊→ZERO
[3]   B1:CREWRANKLIST[CREWRANK[OBJ];] CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:CREWRANK[OBJ]←CREWRANKDEF◊→B1
     ∇


    ∇CUTPICTURE∇


     ∇R←DEF CVTDATE T;H
[1]   R←DEF CVTTIME T
[2]   H←(R[;ONE]≠'*')/ι(ρR)[ONE]◊R[H;(ONE+(ρR)[TWO])-ιSIX]←' '◊R←DLBS3 R
     ∇


     ∇R←DEF CVTDATE1 T;H
[1]   R←DEF CVTTIME1 T◊→(R[ONE]='*')/ZERO◊R←¯6↓R
     ∇


     ∇R←DEF CVTNUM NUMS;C
[1]   R←⍕((ρ,NUMS),ONE)ρNUMS◊C←(NUMS∊DEF)/ιρ,NUMS◊→(ZERO=ρC)/ZERO
[2]   R←(0 11⌈ρR)↑R◊R[C;]←' '◊R[C;ι11]←((ρC),11)ρ'**NO INFO**'
```

```
∇


∇R←DEF CVTNUM1 NUMS;C
[1]  →(NUMS=DEF)/B1◊R←⍕NUMS◊→ZERO
[2]  B1:R←'**NO INFO**'
∇


∇R←DEF CVTTIME T;DAY;YR;RD;F;M
[1]  DEF←DEF≠,T◊T←DEF/T◊RD←ρT◊→(ZERO=RD)/B1◊T←⍉0 365.25 24 60⊤T
[2]  YR←',',⍕1900+T[;,ONE]◊F←¯12 0↓(12×ZERO=FOUR|T[;ONE])∘RD/MOS
[3]  M←ONE++⌿((12,(ρT)[ONE])ρ⍉ONE+⌊T[;,TWO])>F
[4]  DAY←' ',(RD,NTWO)↑⍕ONE+⌊T[;,TWO]-(RD,ONE)ρ1 1⍉(ZERO⍪F)[M;]
[5]  R←DEF⍀MONTHS[M;],DAY,YR,' ',((RD,NTWO)↑⍕T[;,3]),':',(RD,NTWO)↑⍕T[;,4]+100
[6]  DEF←(~DEF)/⍳ρDEF◊R[DEF;⍳11]←((ρDEF),11)ρ'**NO INFO**'◊→ZERO
[7]  B1:R←((ρDEF),11)ρ'**NO INFO**'
∇


∇R←DEF CVTTIME1 T;DAY;YR;RD;F;M
[1]  →(DEF=T)/B1◊T←,0 365.25 24 60⊤T◊YR←',',⍕1900+T[ONE]
[2]  F←(12-24×ZERO=FOUR|T[ONE])↑,MOS
[3]  M←ONE++/(12ρONE+⌊T[TWO])>F◊DAY←' ',NTWO↑⍕ONE+⌊T[TWO]-(ZERO⍪F)[M]
[4]  R←MONTHS[M;],DAY,YR,' ',(NTWO↑⍕T[THREE]),':',NTWO↑⍕T[FOUR]+100◊→ZERO
[5]  B1:R←'**NO INFO**'
∇


∇DATALOAD W0;W1;W2;W3;W4;W5;W7
[1]  W1←ONE+⌈/ZERO,⎕FNUMS◊W0 ⎕FHTIE W1
[2]  B1:W3←NONE↓⎕FREAD W1◊→(ZERO=ρW3)/B3◊W4←NONE↓⎕FREAD W1◊W5←ONE↑⎕FREAD W1
[3]  W2←NONE↓⎕FREAD W1◊→(W5='N')/B2◊→(W5='F')/B4
[4]  B5:W2←(⎕FI W4)ρW2◊⍎W3,'←W2'◊→B1
[5]  B2:W2←(⎕FI W4)ρ⎕FI W2◊⍎W3,'←W2'◊→B1
[6]  B4:W2←(⎕FI W4)ρW2◊W3←⎕FX W2◊→B1
[7]  B3:⎕FUNTIE W1
∇


∇K←W7 DATASAVE KK;G1;G2;G3;G4;G5;G6;G7;⎕ELX
[1]  ⎕ELX←'⎕ERROR((⎕DM⍳⎕TCNL)-⎕IO)↑⎕DM'
[2]  G4←(ρKK)[ONE]◊G1←ONE+⌈/ZERO,⎕FNUMS◊G2←ONE-(⌽W7)⍳':'◊K←G2↓W7◊G3←⎕LIB K
[3]  K←(((ρG3)[ONE],ρK)ρUPPERCASE K),G3
[4]  G1←⌈/(ρW7),ONE↓ρK◊→((((ρK)[ONE],G1)↑K) FIND (ONE,G1)ρG1↑UPPERCASE W7)/B5
[5]  B7:W7 ⎕FHCREATE G1◊K←ONE◊G5←⎕TCNL◊G6←G5,'C',G5◊G7←G5,'N',G5
[6]  B1:G2←DLBS KK[K;]◊→(3=⎕NC G2)/B4◊G3←⍎G2◊→(82≠⎕DR G3)/B2
[7]  (G2,G5,(⍕ρG3),G6,(,G3),G5) ⎕FAPPEND G1◊K←K+ONE◊→(K≤G4)/B1◊→B8
[8]  B2:(G2,G5,(⍕ρG3),G7,(⍕,G3),G5) ⎕FAPPEND G1◊K←K+ONE◊→(K≤G4)/B1◊→B8
[9]  B4:G3←⎕CR G2◊(G2,G5,(⍕ρG3),G5,'F',G5,(,G3),G5) ⎕FAPPEND G1◊K←K+ONE
[10] →(K≤G4)/B1
[11] B8:⎕FUNTIE G1◊→ZERO
[12] B5:W7 ⎕FHTIE G1◊W7 ⎕FERASE G1◊→B7
∇


∇DELETEMENU∇
```

```
     ∇R←DLBS A
[1]   R←(NONE+A⍳' ')↑A
     ∇


     ∇R←DLBS2 A
[1]   R←(-+/∧\' '=⌽A)↓A
     ∇


     ∇R←DLBS3 A
[1]   R←(ZERO,-+/∧\⌽∧≠' '=A)↓A
     ∇


     ∇R←DLBS4 A
[1]   R←(-+/∧\' '=⌽A)↓A⋄R←(+/∧\' '=R)↓R
     ∇


    ⍫DRAWLINE⍫


    ⍫DRAWMENUBAR⍫


     ∇DRAWMFIVEICON;K;H;G;KK
[1]   K←PICTFRAME KK←MFIVEICON2⋄H←⌈GRAYPAT+TWO⋄69 137 184 518 DRAWPICTURE LOGO
[2]   K←172 5,172 5+(K[THREE]-K[ONE]),K[FOUR]-K[TWO]
[3]   K DRAWPICTURE KK⋄→ZERO
     ∇


     ∇A DRAWMSG B
[1]   →(B[ONE]=B[TWO])/B1⋄B←B[TWO]
[2]   B2:A←NONE↓A⋄→(B<TEXTWIDTH A)/B2
[3]   B1:DRAWTEXT A
     ∇


    ⍫DRAWPICTURE⍫


    ⍫DRAWTEXT⍫


     ∇ED R;K;G1;G2;G3;G4;G5;L;M;N;M1;RC;E;FL;LIST;MSG;S;SIZE;HS;UN;UN2;P
[1]   EDIT R
     ∇


     ∇Q←ED2;SROW;SCOL;C1;C2;G1;K
[1]   Q←ONE⋄→(ZERO=MODE)/B3⋄SROW←⌈((NTWO+M[ONE])-S1)÷ELEVEN⋄→(SROW<ONE)/ZERO
[2]   →(SROW>N)/ZERO⋄SCOL←+/FL<LC+(M[TWO]-S2÷FIVE)+SIX⋄C1←SROW+N×SCOL-ONE
[3]   →(AMAX<C1)/ZERO⋄→(C1∊S)/B1⋄→(MODE=ONE)/B5⋄S←S,C1
[4]   B2:C1←S2+SEVEN+SIX×ZERO⌈FL[SCOL]-LC
```

```
C2←ONE+S2+SIX×(RC+ONE)⌊FL[SCOL+ONE]-LC
INVERTRECT (¯9+S1+SROW×ELEVEN),C1,(TWO+S1+SROW×ELEVEN),C2⋄→ZERO
B1:→(MODE=ONE)/B4⋄S←(S≠C1)/S⋄→B2
B3:ERRORWINDOW 'PLEASE SELECT REVISE OR DELETE MODE'⋄→ZERO
B4:ERRORWINDOW 'YOU CANNOT SELECT A DELETED ENTRY'⋄→ZERO
B5:G1←ONE REPLYWINDOW 'ENTER REVISION FOR ',(DLBS2 ,LIST[C1;]),':  '
→(ZERO=⍴G1)/ZERO⋄LIST←DLBS3 LIST[⍳C1-ONE;] APPEND G1 APPEND (C1,ZERO)↓LIST
UN2 ⎕PUTBITS UN⋄Q←ZERO
∇


∇Q←ED2A;SROW;SCOL;C1;C2;G1;K;G2;G3;G4;H;S3
H←(⍴LIST)[TWO]⋄Q←ONE⋄SROW←IZ⍴⌈((NTWO+M[ONE])-S1)÷ELEVEN
→(SROW<ONE)/ZERO⋄→(SROW>N)/ZERO⋄SCOL←+/FL<LC+(M[TWO]-S2+FIVE)÷SIX
C1←IZ⍴SROW+N×SCOL-ONE⋄S3←⍎RPROG⋄→(ZERO=⍴S3)/B10
→(AMAX<C1)/ZERO⋄K←,LIST[C1;]⋄G2←(ONE-(⌽K)⍳'/')↓K
G3←S2+SEVEN+SIX×ZERO⌈FL[SCOL]-LC⋄C2←ONE+S2+SIX×(RC+ONE)⌊FL[SCOL+ONE]-LC
G4←(¯10+S1+SROW×ELEVEN),G3,(ONE+S1+SROW×ELEVEN),C2⋄INVERTRECT G4
B3:G1←4 1 0 0 REPLYWINDOW S3⋄→(OPT=2 3 6)/B6,B2,B5⋄S[C1]←G1
G1←(⍕G1),' '⋄G1←G2,(-(⍴G1)⌈H-⍴G2)↑G1⋄→((⍴G1)>ONE+H)/B1
B8:OUT[SROW;FL[SCOL]+⍳H]←LIST[C1;]←H↑G1
K←S1+NONE+ELEVEN×SROW⋄ERASERECT ¯9 3 2 ¯1+K,S2,K,S4
MOVETO K,EIGHT+S2⋄DRAWTEXT RC↑LC↓,OUT[SROW;]⋄→ZERO
B1:LIST←DLBS3 LIST[⍳C1-ONE;] APPEND G1 APPEND (C1,ZERO)↓LIST
UN2 ⎕PUTBITS UN⋄Q←ZERO⋄→ZERO
B6:INVERTRECT G4⋄→ZERO
B2:G1←((NONE+H)↑G2),'*'⋄S[C1]←NONE⋄→B8
B5:HELPWINDOW HELPMSG⋄→B3
B10:ERRORWINDOW MSG3⋄→ZERO
∇


∇S←C EDA R;K;G1;G2;G3;G4;G5;L;M;N;M1;RC;E;FL;LIST;MSG;S;SIZE;HS;RPROG;P2;C1
S←C EDITA R
∇


∇S←C EDB R;K;G1;G2;G3;G4;G5;L;M;N;M1;RC;E;FL;LIST;MSG;S;SIZE;HS;RPROG;P2;C1
S←C EDITB R
∇


∇EDIT Q;AMAX;S1;S2;S3;S4;OUT;LC;A;ULC;B;RECT;MS;CR;TR;C;OUT2;MODE;C1
N←Q⍳' '⋄MSG←' ',(N↓Q),' '⋄LIST←⍎Q←N↑Q⋄AMAX←(⍴LIST)[ONE]⋄N←18⋄ULC←60 100
TEXTSIZE 12⋄TEXTFONT ZERO⋄TEXTFACE ZERO⋄MS←TEXTWIDTH MSG⋄MODE←ZERO⋄P←⍳13
C1←L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L2,L13⋄LC←ZERO⋄S←IZ
B40:OUT←18 0⍴IZ⋄K←ZERO
B31:OUT←OUT ADJOIN2 (DLBS3 LIST[K+⍳N⌊AMAX-K;]),AV125⋄K←K+N⋄→(AMAX>K)/B31
OUT←0 ¯1↓OUT⋄→(20<(⍴OUT)[TWO])/B45⋄OUT←18 20↑OUT
B45:OUT2←OUT⋄FL←NONE,((OUT[ONE;]=AV125)/⍳(⍴OUT)[TWO]),ONE+(⍴OUT)[TWO]
SIZE←11 6×1 2+⍴OUT⋄SIZE[2]←SIZE[2]⌊246⋄RC←⌊(¯9+SIZE[TWO])÷SIX
B4:S1←ULC[ONE]⋄S2←ULC[TWO]⋄S3←S1+SIZE[ONE]⋄S4←S2+SIZE[TWO]
RECT←S1,S2,S3,S4⋄B←¯18 0 18 168+RECT⋄UN←TWO↑B⋄UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
K←⎕EX 'UN2'⋄UN2←⎕GETBITS UN⋄ERASERECT B⋄TR←¯18 0 1 168+S1,S2,S1,S4
CR←¯14 10 ¯3 22+S1,S2,S1,S2⋄G3←¯1 0 18 18+S3,S2,S3,S2
G3 DRAWPICTURE LEFTARROW⋄G4←¯1 ¯18 18 0+S3,S4,S3,S4
G4 DRAWPICTURE RIGHTARROW⋄K←(S1-15)+TWO×⍳SIX
K←⍉4 6⍴(K-ONE),(SIX⍴TWO+S2),K,SIX⍴S4+165⋄RECT←RECT,5 4⍴CR,TR,G3,G4
FRAMERECT K⍪B⍪RECT⋄G1←⌊PFIVE×168+S4+S2-MS⌊123+S4-S2⋄TEXTSIZE 12
K←(S1-17),G1,(S1-ONE),168+S4+S2-G1
```

```
ERASERECT K⍪3 4⍴(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
MOVETO (S1-THREE),G1⋄TEXTFONT ZERO⋄MSG DRAWMSG MS,MS⌊123+S4-S2⋄TEXTNORMAL
GRAYPAT FILLRECT 0 18 17 ¯18+S3,S2,S3,S4⋄K←¯2 ¯2 2 2 2 2⋄TEXTFACE ONE
C←26 20+S1,S4⋄A←(C+0 ¯5),(C+15 55),15 15⋄E←A,A+K⋄MOVETO 11 10+A[1 2]
DRAWTEXT 'REVISE'⋄RECT←RECT⍪1 4⍴A⋄A←A+0 75 0 75 0 0⋄E←E,A,A+K
MOVETO 11 9+A[1 2]⋄DRAWTEXT 'DELETE'⋄RECT←RECT⍪1 4⍴A
A←A+23 ¯60 23 ¯17 0 0⋄E←E,A,A+K⋄MOVETO 11 8+A[1 2]
DRAWTEXT 'ADD NEW ENTRY'⋄RECT←RECT⍪1 4⍴A⋄A←A+23 0 23 0 0 0⋄E←E,A,A+K
MOVETO 11 20+A[1 2]⋄DRAWTEXT 'DELETE ALL'⋄RECT←RECT⍪1 4⍴A
A←A+23 0 23 0 0 0⋄E←E,A,A+K⋄MOVETO 11 14+A[1 2]⋄DRAWTEXT 'UNDELETE ALL'
RECT←RECT⍪1 4⍴A⋄A←A+23 ¯15 23 ¯58 0 0⋄E←E,A,A+K⋄MOVETO 11 16+A[1 2]
DRAWTEXT 'DONE'⋄RECT←RECT⍪1 4⍴A⋄A←A+0 75 0 75 0 0⋄E←E,A,A+K
MOVETO 11 10+A[1 2]⋄DRAWTEXT 'CANCEL'⋄RECT←RECT⍪1 4⍴A⋄TEXTNORMAL
FRAMEROUNDRECT 14 6⍴E⋄TEXTFACE ZERO
→(ONE≠MODE)/B42⋄INVERTROUNDRECT RECT[SEVEN;],15 15⋄→B14
B42:→(TWO≠MODE)/B14⋄INVERTROUNDRECT RECT[EIGHT;],15 15
B14:MOVETO 10 8+S1,S2⋄DRAWTEXT (N,RC)↑(ZERO,LC)↓OUT⋄ZERO IR RC
B18:G5←(RC×S4-S2+36)+TWO×(⍴OUT)[TWO]
G2←S2+N+((LC+PFIVE×RC)+(⍴OUT)[TWO])×S4-S2+36
G5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5⋄ERASERECT G5⋄FRAMERECT G5
RECT[SIX;]←G5
B2:M←⎕GETKEY⋄→(THREE≠⍴M)/B2⋄→(TWO=M[ONE])/B2⋄M←M[2 3]
L←ONE↑(M PTINRECT RECT)/P⋄→(ZERO=L)/B2⋄→C1[L]
B3:⎕SOUND BEEP⋄→B2
L1:→BUTTON/L1⋄K←ED2⋄→K/B2⋄→B40
L2:→BUTTON/L2⋄→(~GETMOUSE PTINRECT RECT[L;])/B2⋄UN2 ⎕PUTBITS UN
⍎Q,'←DLBS3 LIST ELIM S'⋄→ZERO
L3:M1←B⋄PENPAT GRAYPAT⋄PENMODE TEN
B7:→(~BUTTON)/B5⋄FRAMERECT M1⋄M1←B+FOUR⍴(0 0⌈GETMOUSE)-M⋄FRAMERECT M1⋄→B7
B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
B30:FRAMERECT M1⋄UN2 ⎕PUTBITS UN⋄PENPAT BLACKPAT⋄PENMODE EIGHT⋄→B4
L4:→(LC≤ZERO)/B3⋄HS←(⌽(LC-ONE)↑OUT[ONE;])⍳AV125⋄G3←LC⌊HS
SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO⋄LC←LC-G3
MOVETO 10 8+S1,S2⋄DRAWTEXT (N,G3⌊RC)↑(ZERO,LC)↓OUT
ZERO IR RC⌊G3⋄GRAYPAT FILLRECT G5⋄→B18
L5:G3←(⍴OUT)[TWO]-RC⋄→(LC≥G3)/B3⋄HS←((RC+LC+ONE)↓OUT[ONE;])⍳AV125⋄G3←HS⌊G3
SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(¯6×G3),ZERO⋄LC←LC+G3
MOVETO (S1+TEN),S2+EIGHT+SIX×RC-G3⋄DRAWTEXT (N,-G3)↑(N,RC)↑(ZERO,LC)↓OUT
(ZERO⌈RC-G3) IR RC⋄GRAYPAT FILLRECT G5⋄→B18
L6:M1←G5⋄PENPAT GRAYPAT⋄PENMODE TEN⋄K←ZERO
B23:→(~BUTTON)/B24⋄FRAMERECT M1
K←(18+S2-G5[2])⌈(S4-18+G5[4])⌊ONE↓GETMOUSE-M⋄M1←G5+4⍴0,K⋄FRAMERECT M1⋄→B23
B24:FRAMERECT M1⋄GRAYPAT FILLRECT G5⋄PENPAT BLACKPAT⋄PENMODE EIGHT
LC←LC+⌊PFIVE+K×(⍴OUT)[TWO]+S4-S2+36⋄ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]⋄→B14
L7:→BUTTON/L7⋄INVERTROUNDRECT (,RECT[L;]),15 15⋄→(MODE≠TWO)/B35
INVERTROUNDRECT (,RECT[L+ONE;]),15 15⋄MODE←ONE⋄→B2
B35:MODE←ONE-MODE⋄→B2
L8:→BUTTON/L8⋄INVERTROUNDRECT (,RECT[L;]),15 15⋄→(MODE≠ONE)/B36
INVERTROUNDRECT (,RECT[L-ONE;]),15 15⋄MODE←TWO⋄→B2
B36:MODE←TWO-MODE⋄→B2
L9:K←ONE REPLYWINDOW 'Enter New Entry:  '⋄→(ZERO=⍴K)/B2⋄LIST←LIST APPEND K
OUT←(N,ONE+(NTWO↑FL)[ONE])↑OUT⋄K←N×ZERO⌈NTWO+⍴FL⋄AMAX←AMAX+ONE
B41:OUT←OUT ADJOIN2 (DLBS3 LIST[K+⍳N⌊AMAX-K;]),AV125⋄K←K+N⋄→(AMAX>K)/B41
OUT←0 ¯1↓OUT⋄→(20<(⍴OUT)[TWO])/B46⋄OUT←18 20↑OUT
B46:OUT2←OUT⋄FL←NONE,((OUT[ONE;]=AV125)/⍳(⍴OUT)[TWO]),ONE+(⍴OUT)[TWO]
K←N|AMAX⋄K←(N×K=ZERO)+K
MOVETO (¯1+S1+11×K),8+S2⋄DRAWTEXT RC↑LC↓,OUT[K;]⋄GRAYPAT FILLRECT G5⋄→B18
L10:K←S⋄S←(~(⍳AMAX)∊S)/⍳AMAX⋄ZERO IR RC⋄S←K,S⋄→B2
L11:ZERO IR RC⋄S←IZ⋄→B2
L13:S←IZ⋄UN2 ⎕PUTBITS UN⋄→ZERO
∇
```

```
    ∇S←P EDITA Q;AMAX;S1;S2;S3;S4;OUT;LC;A;ULC;B;RECT;MS;CR;TR;C;OUT2;UN;UN2;ML
[1]   N←Q⍳' '◇MSG←N↓Q◇LIST←⍎Q←N↑Q◇AMAX←(⍴LIST)[ONE]◇N←18◇ULC←60 100
[2]   K←MSG⍳' '◇RPROG←K↑MSG◇MSG←' ',(K↓MSG),' '◇S←P
[3]   TEXTSIZE 12◇TEXTFONT ZERO◇TEXTFACE ZERO◇MS←TEXTWIDTH MSG◇ML←TEXTWIDTH MSG2
[4]   P2←⍳SEVEN◇C1←L1,L2,L13,L3,L4,L5,L6◇LC←ZERO
[5]   B40:OUT←18 0⍴IZ◇K←ZERO
[6]   B31:OUT←OUT ADJOIN2 LIST[K+⍳N⌊AMAX-K;],AV125◇K←K+N◇→(AMAX>K)/B31
[7]   OUT←0 ¯1↓OUT◇→(20<(⍴OUT)[TWO])/B45◇OUT←18 20↑OUT
[8]   B45:OUT2←OUT◇FL←ZERO,((OUT[ONE;]=AV125)/⍳(⍴OUT)[TWO]),ONE+(⍴OUT)[TWO]
[9]   SIZE←11 6×1 2+⍴OUT◇SIZE[TWO]←(ML+TEN)⌈(MS+50)⌈SIZE[2]⌊360
[10]  RC←⌊(¯9+SIZE[TWO])÷SIX
[11]  B4:S1←ULC[ONE]◇S2←ULC[TWO]◇S3←S1+SIZE[ONE]◇S4←S2+SIZE[TWO]
[12]  RECT←S1,S2,S3,S4◇B←¯36 0 18 0+RECT◇UN←TWO↑B◇UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
[13]  K←⎕EX 'UN2'◇UN2←⎕GETBITS UN◇ERASERECT B◇TR←¯36 0 ¯17 0+S1,S2,S1,S4
[14]  CR←¯32 10 ¯21 22+S1,S2,S1,S2◇G3←¯1 0 18 18+S3,S2,S3,S2
[15]  G3 DRAWPICTURE LEFTARROW◇G4←¯1 ¯18 18 0+S3,S4,S3,S4
[16]  G4 DRAWPICTURE RIGHTARROW◇K←(S1-33)+TWO×⍳SIX◇C←¯32 ¯16 ¯21 ¯4+S1,S4,S1,S4
[17]  K←⍉4 6⍴(K-ONE),(SIX⍴TWO+S2),K,SIX⍴S4+NTWO◇RECT←RECT⍪6 4⍴CR,C,TR,G3,G4
[18]  FRAMERECT K⍪B⍪RECT◇G1←⌊PFIVE×S4+S2-MS⌊S4-S2+45◇TEXTSIZE 12
[19]  K←(S1-35),G1,(S1-19),(S4+S2-G1),0 ¯1 1 0 0 0 1 1+C[1 2 3 2 1 4 3 4]
[20]  ERASERECT 6 4⍴K,(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
[21]  MOVETO (S1-21),G1◇TEXTFONT ZERO◇MSG DRAWMSG MS,MS⌊S4-S2+45
[22]  MOVETO (S1-THREE),⌊PFIVE×S4+S2-ML⌊S4-S2◇MSG2 DRAWMSG ML,ML⌊S4-S2
[23]  TEXTNORMAL◇GRAYPAT FILLRECT 2 4⍴(C+1 1 ¯1 ¯1),0 18 17 ¯18+S3,S2,S3,S4
[24]  B14:MOVETO 10 8+S1,S2◇DRAWTEXT (N,RC)↑(ZERO,LC)↓OUT
[25]  B18:G5←(RC×S4-S2+36)÷TWO×(⍴OUT)[TWO]
[26]  G2←S2+N+((LC+PFIVE×RC)÷(⍴OUT)[TWO])×S4-S2+36
[27]  G5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◇ERASERECT G5◇FRAMERECT G5
[28]  RECT[SEVEN;]←G5
[29]  B2:M←⎕GETKEY◇→(THREE≠⍴M)/B2◇→(TWO=M[ONE])/B2◇M←M[2 3]
[30]  L←ONE↑(M PTINRECT RECT)/P2◇→(ZERO=L)/B2◇→C1[L]
[31]  B3:⎕SOUND BEEP◇→B2
[32]  L1:→BUTTON/L1◇K←ED2A◇→K/B2◇→B40
[33]  L2:→BUTTON/L2◇→(∼GETMOUSE PTINRECT RECT[L;])/B2◇UN2 ⎕PUTBITS UN◇→ZERO
[34]  L3:M1←B◇PENPAT GRAYPAT◇PENMODE TEN
[35]  B7:→(∼BUTTON)/B5◇FRAMERECT M1◇M1←B+FOUR⍴(0 0⌈GETMOUSE)-M◇FRAMERECT M1◇→B7
[36]  B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
[37]  B30:FRAMERECT M1◇UN2 ⎕PUTBITS UN◇PENPAT BLACKPAT◇PENMODE EIGHT◇→B4
[38]  L4:→(LC≤ZERO)/B3◇HS←(⌽(LC-ONE)↑OUT[ONE;])⍳AV125◇G3←LC⌊HS
[39]  SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO◇LC←LC-G3
[40]  MOVETO 10 8+S1,S2◇DRAWTEXT (N,G3⌊RC)↑(ZERO,LC)↓OUT◇GRAYPAT FILLRECT G5
[41]  →B18
[42]  L5:G3←(⍴OUT)[TWO]-RC◇→(LC≥G3)/B3◇HS←((RC+LC+ONE)↓OUT[ONE;])⍳AV125◇G3←HS⌊G3
[43]  SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(¯6×G3),ZERO◇LC←LC+G3
[44]  MOVETO (S1+TEN),S2+EIGHT+SIX×RC-G3◇DRAWTEXT (N,-G3)↑(N,RC)↑(ZERO,LC)↓OUT
[45]  GRAYPAT FILLRECT G5◇→B18
[46]  L6:M1←G5◇PENPAT GRAYPAT◇PENMODE TEN◇K←ZERO
[47]  B23:→(∼BUTTON)/B24◇FRAMERECT M1
[48]  K←(18+S2-G5[2])⌈(S4-18+G5[4])⌊ONE↓GETMOUSE-M◇M1←G5+4⍴0,K◇FRAMERECT M1◇→B23
[49]  B24:FRAMERECT M1◇GRAYPAT FILLRECT G5◇PENPAT BLACKPAT◇PENMODE EIGHT
[50]  LC←LC+⌊PFIVE+K×(⍴OUT)[TWO]÷S4-S2+36◇ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◇→B14
[51]  L13:S←P◇UN2 ⎕PUTBITS UN◇→ZERO
    ∇
```

```
    ∇S←P EDITB Q;AMAX;S1;S2;S3;S4;OUT;LC;A;ULC;B;RECT;MS;TR;OUT2;UN;UN2;ML
[1]   N←Q⍳' '◇MSG←N↓Q◇LIST←⍎Q←N↑Q◇AMAX←(⍴LIST)[ONE]◇N←18◇ULC←60 100
[2]   K←MSG⍳' '◇RPROG←K↑MSG◇MSG←' ',(K↓MSG),' '◇S←P
[3]   TEXTSIZE 12◇TEXTFONT ZERO◇TEXTFACE ZERO◇MS←TEXTWIDTH MSG◇ML←TEXTWIDTH MSG2
[4]   P2←⍳SEVEN◇C1←L1,L3,L4,L5,L6,L8,L9◇LC←ZERO
[5]   B40:OUT←18 0⍴IZ◇K←ZERO
[6]   B31:OUT←OUT ADJOIN2 (DLBS3 LIST[K+⍳N⌊AMAX-K;]),AV125◇K←K+N◇→(AMAX>K)/B31
[7]   OUT←0 ¯1↓OUT◇→(20<(⍴OUT)[TWO])/B45◇OUT←18 20↑OUT
```

```
B45:OUT2←OUT◊FL←NONE,((OUT[ONE;]=AV125)/ι(ρOUT)[TWO]),ONE+(ρOUT)[TWO]
SIZE←11 6×1 2+ρOUT◊SIZE[TWO]←(ML+90)⌈(MS+50)⌈SIZE[2]⌊346
RC←⌊(¯9+SIZE[TWO])÷SIX
B4:S1←ULC[ONE]◊S2←ULC[TWO]◊S3←S1+SIZE[ONE]◊S4←S2+SIZE[TWO]
RECT←S1,S2,S3,S4◊B←¯36 0 18 0+RECT◊UN←TWO↑B◊UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
K←⎕EX 'UN2'◊UN2←⎕GETBITS UN◊ERASERECT B◊TR←¯36 0 ¯17 0+S1,S2,S1,S4
G3←¯1 0 18 18+S3,S2,S3,S2
G3 DRAWPICTURE LEFTARROW◊G4←¯1 ¯18 18 0+S3,S4,S3,S4
G4 DRAWPICTURE RIGHTARROW◊K←(S1-33)+TWO×ιSIX
K←⍉4 6ρ(K-ONE),(SIXρTWO+S2),K,SIXρS4+NTWO◊RECT←RECT,[ONE]4 4ρTR,G3,G4
FRAMERECT K,[ONE]B,[ONE]RECT◊G1←⌊PFIVE×S4+S2-MS⌊S4-S2+45◊TEXTSIZE 12
ERASERECT(S1-35),G1,(S1-19),(S4+S2-G1)◊MOVETO (S1-21),G1◊TEXTFONT ZERO
MSG DRAWMSG MS,MS⌊S4-S2+45◊MOVETO (S1-FOUR),S2+EIGHT◊DRAWTEXT MSG2
K←¯4 ¯60+S1,S4◊MOVETO K◊DRAWTEXT 'YES    NO'◊TEXTNORMAL
K←2 4ρ¯12 ¯2 3 24 ¯12 29 3 55+8ρK◊RECT←RECT,[ONE]K
FRAMEROUNDRECT K,2 2ρ7 7◊GRAYPAT FILLRECT 0 18 17 ¯18+S3,S2,S3,S4
B14:MOVETO 10 8+S1,S2◊DRAWTEXT (N,RC)↑(ZERO,LC)↓OUT
B18:G5←(RC×S4-S2+36)÷TWO×(ρOUT)[TWO]
G2←S2+N+((LC+PFIVE×RC)÷(ρOUT)[TWO])×S4-S2+36
G5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◊ERASERECT G5◊FRAMERECT G5
RECT[FIVE;]←G5
B2:M←⎕GETKEY◊→(THREE≠ρM)/B2◊→(TWO=M[ONE])/B2◊M←M[2 3]
L←ONE↑(M PTINRECT RECT)/P2◊→(ZERO=L)/B2◊→C1[L]
B3:⎕SOUND BEEP◊→B2
L1:→BUTTON/L1◊K←ED2A◊→K/B2◊→B40
L3:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
B7:→(~BUTTON)/B5◊FRAMERECT M1◊M1←B+FOURρ(0 0⌈GETMOUSE)-M◊FRAMERECT M1◊→B7
B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
B30:FRAMERECT M1◊UN2 ⎕PUTBITS UN◊PENPAT BLACKPAT◊PENMODE EIGHT◊→B4
L4:→(LC≤ZERO)/B3◊HS←(⌽(LC-ONE)↑OUT[ONE;])ιAV125◊G3←LC⌊HS
SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO◊LC←LC-G3
MOVETO 10 8+S1,S2◊DRAWTEXT (N,G3⌊RC)↑(ZERO,LC)↓OUT◊GRAYPAT FILLRECT G5
→B18
L5:G3←(ρOUT)[TWO]-RC◊→(LC≥G3)/B3◊HS←((RC+LC+ONE)↓OUT[ONE;])ιAV125◊G3←HS⌊G3
SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(¯6×G3),ZERO◊LC←LC+G3
MOVETO (S1+TEN),S2+EIGHT+SIX×RC-G3◊DRAWTEXT (N,-G3)↑(N,RC)↑(ZERO,LC)↓OUT
GRAYPAT FILLRECT G5◊→B18
L6:M1←G5◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
B23:→(~BUTTON)/B24◊FRAMERECT M1
K←(18+S2-G5[2])⌈(S4-18+G5[4])⌊ONE↓GETMOUSE-M◊M1←G5+4ρ0,K◊FRAMERECT M1◊→B23
B24:FRAMERECT M1◊GRAYPAT FILLRECT G5◊PENPAT BLACKPAT◊PENMODE EIGHT
LC←LC+⌊PFIVE+K×(ρOUT)[TWO]÷S4-S2+36◊ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◊→B14
L8:S←ONE◊→B30
L9:S←ZERO
B30:→BUTTON/B30◊→(~GETMOUSE PTINRECT RECT[L;])/B2◊UN2 ⎕PUTBITS UN◊→ZERO
∇
```

```
∇R←A ELIM B
  R←(~(ι(ρA)[ONE])ε,B)⌿A
∇
```

```
∇ERASERECT∇
```

```
∇ERRORWINDOW A;OS;K;M;D;UN;UN2;KK;P
 D←22 20 70 480◊UN←22 20 70 484◊KK←ONE
 UN2←⎕GETBITS UN◊ERASERECT D◊FRAMERECT 2 4ρD,24 22 68 478
 FRAMEROUNDRECT 2 6ρ33 385 57 460 15 15 31 383 59 462 17 17
 TEXTSIZE 18◊TEXTFACE 65◊MOVETO 52 407◊DRAWTEXT 'OK'◊⎕SOUND BEEP
 TEXTSIZE NINE◊TEXTFACE ONE◊K←35+FIVE×FOUR-THREE⌊⌈(ρA)÷45
```

```
[6]    B1:MOVETO K,50◊OS←45-(⌽45↑A)⍳' '◊P←OS<ZERO◊OS←OS+45×P◊→((KK=3)∧OS<⍴A)/B5
[7]    DRAWTEXT OS↑A◊KK←KK+ONE◊A←(OS+~P)↓A◊K←K+ELEVEN◊→(ZERO≠⍴A)/B1◊TEXTFACE ZERO
[8]    B2:SHOWCURSOR◊M←⎕GETKEY◊→(THREE≠⍴M)/B2◊→(TWO=M[ONE])/B2
[9]    →(~M[2 3] PTINRECT 33 385 57 460)/B3◊UN2 ⎕PUTBITS UN
[10]   B4:→(ZERO≠⍴⎕GETKEY)/B4◊→ZERO
[11]   B3:SHOWCURSOR◊→BUTTON/B3◊→B2
[12]   B5:DRAWTEXT (41↑A),'...'◊TEXTFACE ZERO◊→B2
     ∇


     ∇A FILEHCREATE B;K
[1]    K←⎕ELX◊⎕ELX←'→B1'
[2]    B2:A ⎕FHCREATE B◊⎕ELX←K◊→ZERO
[3]    B1:⎕ELX←K◊⎕FUNTIE B◊A ⎕FHTIE B◊A ⎕FERASE B◊→B2
     ∇


     ∇R←FILEOPEN P
[1]    B1:R←⎕SFOPEN 'TEXT'◊CLEAR◊→(ZERO=⍴R)/ZERO◊→(P≡(⍴P)↑(ONE-(⌽R)⍳':')↑R)/ZERO
[2]    ERRORWINDOW 'Please Select A File Whose Name Begins With :  ',(¯1↓P)◊→B1
     ∇


     ∇R←A FILESAVE B;G1
[1]    →(~':'∊B)/B2◊B←(B⍳':')↓B
[2]    B2:G1←B⍳'_'
[3]    B1:R←A ⎕SFSAVE B◊CLEAR◊→(ZERO=⍴R)/ZERO◊→((G1↑B)≡G1↑(ONE-(⌽R)⍳':')↑R)/ZERO
[4]    ERRORWINDOW 'Please Begin File Name With :  ',(¯1↓B),'_'◊→B1
     ∇


    ⍒FILLRECT⍒


     ∇R←A FIND B;D
[1]    D←⌈/(⍴A)[TWO],(⍴B)[TWO]◊R←∨⌿(((⍴A)[ONE],D)↑A)∧.=⍉((⍴B)[ONE],D)↑B
     ∇


     ∇R←W FIND3 T
[1]    R←(∨/∧⌿(⍉((⍴T)[ONE],⍴W)⍴NONE+⍳⍴W)⌽W∘.=T)/⍳(⍴T)[ONE]
     ∇


     ∇D FORMGET A
[1]    ERASERECT 70 0 299 507
[2]    ⍎'(73+',A,'FORMSIZE) DRAWPICTURE ',A,'FORM'
[3]    RECT←73+⍎A,'FORMRECTS'◊R2←RECT[;1 5 3 4]
[4]    KW←⍎A,'OPTS'◊→(ZERO≠⍴D)/B1◊C←ONE+A FIND3 COMLIST
[5]    OBJ←(SIX,(A FIND3 COMLIST),1 1) GETINPUT NONE↓RECT[ONE;]
[6]    →(NONE≠ONE↑OBJ)/B3◊OBJ←GETNAME A
[7]    B3:→(ZERO=⍴OBJ)/B2◊⍎A,'FORMFILL OBJ'◊→ZERO
[8]    B1:⍎A,'LIST[D+ONE;] CLIST NONE↓RECT[ONE;]'
[9]    ⍎A,'FORMFILL OBJ←D'◊→ZERO
[10]   B2:RECT←0 5⍴IZ◊R2←0 4⍴IZ◊ERASERECT 70 0 299 507◊DRAWMFIVEICON
     ∇
```

```
∇FRAMERECT∇

∇FRAMEROUNDRECT∇

     ∇R←B GETINP A;ST;G1;G2;G3;LASTP;CURP;E;A1;A2;D;CLASS;NLIST;KK;N;UR;M;A3;AA
[1]   AA←TWO⌈NONE+⌊(A[FOUR]-A[TWO])÷SIX
[2]   L1:OPT←ONE⋄ST←(THREE+⌈PFIVE×A[THREE]+A[ONE]),FIVE+A[TWO]
[3]   A←FOUR↑A⋄ERASERECT 0 0 3 0+A⋄B←,B⋄HIDECURSOR⋄G1←ONE↑B⋄→(6 7=G1)/L23,L42
[4]   G2←~(ONE=G1)×(TWO↑B)[TWO]⋄G3←(3↑B)[3]×ONE=G1⋄C1←¯8 0 2 0⋄C2←13
[5]   R←''⋄LASTP←ST⋄TEXTFACE ONE⋄MOVETO ST⋄DRAWTEXT AV125
[6]   CURP←GETPEN⋄A3←210ρ'→(ONE=ρE←⎕GETKEY)/B4⋄'⋄A2←210ρ'→(ONE=ρE←⎕GETKEY)/B5⋄'
[7]   B1:E←⎕GETKEY⋄→(ONE=ρE)/B4⋄⍎A3⋄ERASERECT C1+ST,CURP
[8]   E←⎕GETKEY⋄→(ONE=ρE)/B5⋄⍎A2⋄MOVETO ST⋄DRAWTEXT AV125⋄→B1
[9]   B4:ERASERECT C1+ST,CURP
[10]  B5:MOVETO ST⋄→(C2=E)/B3⋄→(EIGHT=E)/B2⋄→(E>255)/B1
[11]  KK←AV[E+ONE]⋄→G2/B6⋄→(~KK∊VAL)/B16
[12]  B6:D←KK⋄R←R,D⋄TEXTFACE ZERO⋄DRAWTEXT D
[13]  →(AA≤ρR)/B7⋄ST←GETPEN⋄LASTP←LASTP,ST
[14]  B9:→(G3=ρR)/B3⋄TEXTFACE ONE⋄DRAWTEXT AV125⋄CURP←GETPEN⋄→B1
[15]  B7:SCROLLRECT (¯1 5 2 0+A),¯6 0⋄→(G3=ρR)/B3⋄TEXTFACE ONE⋄→B1
[16]  B10:SCROLLRECT (¯1 5 2 0+A),6 0⋄MOVETO TWO↑LASTP
[17]  TEXTFACE ZERO⋄DRAWTEXT R[TWO+(ρR)-AA]⋄TEXTFACE ONE⋄→B14
[18]  B16:⎕SOUND BEEP⋄→B1
[19]  B2:→(ZERO=ρR)/B16⋄R←NONE↓R⋄ERASERECT C1+(TWO↑NFOUR↑ST,LASTP),ST
[20]  →(AA≤ONE+ρR)/B10⋄LASTP←(TWO⌈NTWO+ρLASTP)↑LASTP
[21]  B14:MOVETO ST←NTWO↑LASTP⋄DRAWTEXT AV125⋄CURP←GETPEN⋄→B1
[22]  B3:UR←UPPERCASE R⋄OPT←ONE+(ZERO=ρR)+(FIVE×UR≡'HELP')+TWO×UR≡'DELETE'
[23]  TEXTFACE ZERO⋄INITCURSOR⋄→((G1≠NINE)∧OPT≠ONE)/ZERO
[24]  →(1 2 3 4 5 8 9=G1)/ZERO,L8,L8,L16,L22,L2,B8
[25]  L2:B←B,(ONE=ρB)/ZERO⋄R←B[TWO] CONVT3 R⋄→(ZERO≠ρ,R)/B21
[26]  ERRORWINDOW 'INVALID TIME'⋄→L1
[27]  B21:(CVTTD R) CLIST A⋄→(THREE>ρB)/ZERO⋄→(B[THREE]≤R)/B20
[28]  ERRORWINDOW 'INVALID TIME.  EARLIEST VALID TIME IS ',,CVTTD B[THREE]⋄→L1
[29]  B20:→(FOUR>ρB)/ZERO⋄→(B[FOUR]≥R)/ZERO
[30]  ERRORWINDOW 'INVALID TIME.  LATEST VALID TIME IS ',,CVTTD B[FOUR]⋄→L1
[31]  L8:R←UR⋄B←B,(ONE=ρB)/NONE⋄R←B[TWO] CONVT2 R,(FOUR×TWO=ONE↑B)↑' 0 0'
[32]  →(ZERO≠ρR)/L12⋄ERRORWINDOW 'INVALID ',(NFOUR+EIGHT×TWO=G1)↑'DATETIME'⋄→L1
[33]  L12:((ZERO,NFIVE×G1=TWO)↓NONE CVTTIME R) CLIST A⋄→(THREE>ρB)/ZERO
[34]  →(B[THREE]≤R)/L14⋄→(THREE=G1)/L13
[35]  ERRORWINDOW 'INVALID DATE.  EARLIEST VALID DATE IS ',,¯1 CVTDATE B[3]⋄→L1
[36]  L13:ERRORWINDOW 'INVALID TIME.  EARLIEST VALID TIME IS ',,¯1 CVTTIME B[3]
[37]  →L1
[38]  L14:→(FOUR>ρB)/ZERO⋄→(B[FOUR]≥R)/ZERO⋄→(THREE=G1)/L15
[39]  ERRORWINDOW 'INVALID DATE.  LATEST VALID DATE IS ',,¯1 CVTDATE B[FOUR]⋄→L1
[40]  L15:E←'INVALID TIME.  LATEST VALID TIME IS ',,¯1 CVTTIME B[FOUR]
[41]  ERRORWINDOW E⋄→L1
[42]  L16:→(~ZERO∊,⎕VI R)ρL17
[43]  ERRORWINDOW 'INVALID ENTRY.  ENTER NUMERICAL VALUES ONLY.'⋄→L1
[44]  L17:R←,⎕FI R⋄(⍕R) CLIST A⋄→(ONE=ρB)/0⋄→(ZERO=B[TWO])/L18⋄→(B[TWO]≥ρR)/L18
[45]  E←'TOO MANY ENTRIES.  ENTER AT MOST ',(⍕B[TWO]),(-ONE=B[TWO])↓' NUMBERS'
[46]  ERRORWINDOW E⋄→L1
[47]  L18:→(TWO=ρB)/ZERO⋄→(ZERO=B[THREE])/L19⋄→(~ZERO∊R=⌊R)/L19
[48]  ERRORWINDOW 'INVALID ENTRY.  ENTER INTEGERS ONLY.'⋄→L1
[49]  L19:→(THREE=ρB)/ZERO⋄→(FOUR=ρB)/L20⋄→(B[FOUR]>B[FIVE])/L21
[50]  L20:→(~ONE∊R<B[FOUR])/L21
[51]  ERRORWINDOW 'INVALID ENTRY.  ENTER VALUE GREATER THAN OR EQUAL TO ',⍕B[4]
[52]  →L1
[53]  L21:→(FOUR=ρB)/ZERO⋄→(~ONE∊R>B[FIVE])/ZERO
[54]  ERRORWINDOW 'INVALID ENTRY.  ENTER VALUE LESS THAN OR EQUAL TO ',⍕B[5]⋄→L1
[55]  L22:R←UR⋄R←TWO-(R≡,'Y')+(R≡'YES')+TWO×(R≡,'N')+R≡'NO'
[56]  →(R=TWO)/B15⋄((ONE,|EIGHT+NFIVE×R)ρ(EIGHT+NFIVE×R)↑'YESNO') CLIST A⋄→ZERO
[57]  B15:ERRORWINDOW 'PLEASE ENTER ''YES'' OR ''NO''.'⋄→L1
```

```
L23:CLASS←B[TWO]◇MAX←(THREE↑B)[THREE]◇NEW←NONE↑FOUR↑B◇B←FOUR↓B
→(CLASS=2 3 4 5)/B11,B12,B13,B42
NLIST←' ',(UPPERCASE A1←1 0↓CREWLIST APPEND RR←CREWN),' '◇N←NN←CREWNN◇→L24
B11:NLIST←' ',(UPPERCASE A1←1 0↓JOBLIST APPEND RR←JOBN),' '◇N←NN←JOBNN
→L24
B12:NLIST←' ',(UPPERCASE A1←1 0↓MANLIST APPEND RR←MANN),' '◇N←NN←MANNN
→L24
B13:NLIST←' ',(UPPERCASE A1←1 0↓TOOLLIST APPEND RR←TOOLN),' '◇N←NN←TOOLNN
→L24
B42:NLIST←' ',(UPPERCASE A1←1 0↓STOWLIST APPEND RR←STOWN),' '◇N←NN←STOWNN
L24:RR←ONE↑ρRR◇R←IZ◇NLIST←(G3←(~NN∊B))≠NLIST◇NN←G3/NN
V4←((ιρNN)=NNιNN)/NN◇MAX←((MAX≠ZERO)↑MAX),(MAX=ZERO)↑ρV4
L25:KK←B◇B←NINE◇M←R◇→L1
B8:COMAND←UR◇R←M◇B←KK◇→(OPT=SIX)/L39◇→(OPT≠ONE)/ZERO
→(NEW∧COMAND≡'NEW')/L50◇→((COMAND≡'ALL')∧MAX≥ρV4)/L37
COMAND←' ' CHARMAT COMAND◇COMAND←(~NULL FIND COMAND)≠COMAND
L26:K←ONE
L27:RCOM←ONE↑ρCOMAND◇→(K>RCOM)/L36◇KK←K
COMM←' ',DLBS COMAND[K;]◇→(ONE↑⎕VI COMM)/L32
OBJ←COMM FIND3 NLIST◇→((ZERO=ρOBJ)∧(ZERO=ρR),ONE)/L35,L36◇G1←NLIST[OBJ;]
L31:KK←KK+ONE◇→(RCOM<KK)/L33◇G3←(' ',DLBS COMAND[KK;]) FIND3 G1
→(ZERO=ρG3)/L33◇OBJ←OBJ[G3]◇G1←G1[G3;]◇→L31
L32:G3←⎕FI COMM◇→((G3≠⌊G3)∨G3<ONE)/L28◇→(~G3∊V4)/L28◇KK←KK+ONE◇OBJ←G3◇→L34
L28:COMAND←COMAND ELIM K◇→L27
L33:G3←OBJ◇OBJ←NN[G3]◇OBJ←((ιρOBJ)=OBJιOBJ)/OBJ◇→(ONE≠ρOBJ)/L40
L34:R←R,OBJ◇COMAND←COMAND ELIM NONE+K+ιKK-K◇→(MAX=ρR)/L43◇→L27
L35:K←K+ONE◇→L27
L36:R←((ιρR)=RιR)/R◇→(ZERO≠ρR)/L43◇→(MAX<ρV4)/L38
L37:R←V4◇→L43
L38:ERRORWINDOW 'I DON''T UNDERSTAND ''',(NONE↓BL ,COMAND,' '),''''◇→L25
L39:MSG2←'PLEASE SELECT FROM THE FOLLOWING LIST:'◇MSG←''
LIST←(-RR,ZERO)↓NLIST◇R←CHOICE IZ◇OPT←ONE+ZERO=ρR◇R←NN[R]◇→L43
L40:G2←(+/E)-+/∧\⌽E←G1=' '◇G2←(G2=KK-K)/ιρG2◇→(ZERO=ρG2)/L41
OBJ←NN[G3[G2]]◇OBJ←((ιρOBJ)=OBJιOBJ)/OBJ◇→(ONE=ρOBJ)/L34
L41:MSG2←'AMBIGUOUS REFERENCE.  PLEASE SELECT THE CORRECT ENTR'
MSG2←MSG2,(THREE+NFOUR×ONE=MAX)↑'IESY'◇MSG←''◇LIST←NLIST[NNιOBJ;]
E←CHOICE IZ◇OBJ←OBJ[E]◇→L34
L42:NLIST←' ',((A1←⍎VAL,'LIST') APPEND RR←⍎VAL,'N'),' '◇N←NN←⍎VAL,'NN'
MAX←(TWO↑B)[TWO]◇NEW←(THREE↑B)[THREE]◇B←THREE↓B◇→L24
L43:A1[NιR;] CLIST A◇R←N⍳R◇→ZERO
L50:R←NONE◇→ZERO
∇
```

```
∇R←B GETINPUT A;MAX;V4;NN;OBJ;MSG;RCOM;MSG2;COMM;COMAND;NEW;LIST;RR;K;C1;C2
 R←B GETINP A
∇
```

∇GETMOUSE∇

```
∇R←GETNAME A;D;F;B
 R←ONE REPLYWINDOW 'Enter New Name:'◇B←ONE↓A
 →(ZERO=ρR)/ZERO◇⍎B,'LIST←',B,'LIST APPEND R'
 F←(⍎'(ιρ',B,'NN)∊',B,'NN')ιZERO
 R←D+⍎'ρ',B,'NN←(D↓,',B,'NN),F,(D←-ONE↑ρ',B,'N)↑',B,'NN'
 ⍎B,'LIST[R+ONE;] CLIST NONE↓RECT[ONE;]'
 →(B2,B3,B4,B5,B6)[A FIND3 COMLIST]
 B2:CREWMS←CREWMS,CREWMSDEF◇CREWRANK←CREWRANK,CREWRANKDEF
 CREWHT←CREWHT,CREWHTDEF◇CREWBD←CREWBD,CREWBDDEF◇CREWAS←CREWAS,CREWASDEF◇→0
 B3:JOBSIZE←JOBSIZE,JOBSIZEDEF◇JOBMEMORY←JOBMEMORY,JOBMEMORYDEF
```

```
[10]  JOBTRANS←JOBTRANS,JOBTRANSDEF⋄JOBPOWER←JOBPOWER,JOBPOWERDEF
[11]  JOBMULTIPLEX←JOBMULTIPLEX,JOBMULTIPLEXDEF⋄JOBAUDIO←JOBAUDIO,JOBAUDIODEF
[12]  PERFDATA1←PERFDATA1,-F⋄PERFDATA2←PERFDATA2,JOBTIMEDEF⋄→ZERO
[13]  B4:MANPOWER←MANPOWER,MANPOWERDEF⋄MANMEMORY←MANMEMORY,MANMEMORYDEF
[14]  MANTRANS←MANTRANS,MANTRANSDEF⋄MANMULTIPLEX←MANMULTIPLEX,MANMULTIPLEXDEF
[15]  MANAS←MANAS,MANASDEF⋄MANCREW←MANCREW,-F⋄MANJOB←MANJOB,-F⋄→ZERO
[16]  B5:TOOLUSE1←TOOLUSE1,-F⋄TOOLUSE2←TOOLUSE2,ZERO
[17]  TOOLVOL←TOOLVOL,TOOLVOLDEF⋄TOOLSTOW←TOOLSTOW,-F⋄TOOLHIST←TOOLHIST,-F
[18]  TOOLSTAT←TOOLSTAT,-F⋄TOOLUSEMODS←TOOLUSEMODS,-F
[19]  TOOLUSERS←TOOLUSERS,-F⋄TOOLUSEX←TOOLUSEX,-F⋄→ZERO
[20]  B6:STOWVOL←STOWVOL,STOWVOLDEF⋄STOWX←STOWX,STOWXYZDEF
[21]  STOWMOD←STOWMOD,STOWMODDEF⋄STOWTHETA←STOWTHETA,STOWXYZDEF
     ∇
```

```
   ∇GETPEN∇
```

```
     ∇HCL INS;K;J;FNS;VAR;TEXT;LINE;IDLIST;R;G;G1;M
[1]   →(INS≡'ALL')/L8⋄VAR←0 0⍴''⋄FNS←' ' CHARMAT ,INS⋄→L9
[2]   L8:⎕EX 'INS'⋄VAR←⎕NL TWO⋄FNS←60 0↓⎕NL THREE
[3]   L9:'ALIGN PRINTER PAPER; THEN PRESS RETURN'⋄⎕SOUND BEEP
[4]   L7:→(ONE≠⍴⎕GETKEY)/L7⋄⎕PRSELECT⋄K←ZERO⍝⋄6⍴⎕TCNL
[5]   R←'Z←IDLIST L;W;B',⎕TCNL,'W←80-(ONE↓⍴L←'' '',L)|80'
[6]   R←R,' ⋄ B←(1↑(0,W)⊤×/⍴L)+×|×/⍴L ⋄ Z←(B,W)⍴(B×W)↑,L ⋄ Z←,Z,⎕TCNL '
[7]   0 0⍴⎕FX ⎕TCNL CHARMAT R
[8]   ⍝'      )FNS',⎕TCNL,(IDLIST FNS),2⍴⎕TCNL
[9]   L1:K←K+ONE⋄→((⍴FNS)[ONE]<K)/L6⋄TEXT←⎕VR FNS[K;]
[10]  →(ZERO=⍴TEXT)/L4⋄TEXT,3⍴⎕TCNL⋄→L1
[11]  L4:'     ∇',(DLBS FNS[K;]),'∇'⋄''⋄''⋄→L1
[12]  L6:→(ZERO∊⍴VAR)/L10⋄'      )VARS',⎕TCNL,(IDLIST VAR),2⍴⎕TCNL
[13]  L10:⎕←⎕TCFF⋄⎕PRUNSELECT⋄⎕←'**DONE**'⋄⎕SOUND BEEP
     ∇
```

```
     ∇HELPWINDOW A;OS;K;M;D;UN;UN2;KK;P;H;G1
[1]   A←⎕TCNL CHARMAT A
[2]   H←(⍴A)[ONE]⌈THREE⋄G1←22+16×⌈(18+11×H)+16⋄D←22 20,G1,480⋄UN←22 20,G1,484
[3]   KK←ONE⋄UN2←⎕GETBITS UN⋄ERASERECT D⋄FRAMERECT 2 4⍴D,24 22,(G1-TWO),478
[4]   FRAMEROUNDRECT 2 6⍴33 385 57 460 15 15 31 383 59 462 17 17
[5]   TEXTSIZE 18⋄TEXTFACE 65⋄MOVETO 52 407⋄DRAWTEXT 'OK'⋄⎕SOUND BEEP
[6]   TEXTNORMAL⋄K←40
[7]   B1:MOVETO K,50⋄DRAWTEXT A
[8]   B2:SHOWCURSOR⋄M←⎕GETKEY⋄→(THREE≠⍴M)/B2⋄→(TWO=M[ONE])/B2
[9]   →(~M[2 3] PTINRECT 33 385 57 460)/B3⋄UN2 ⎕PUTBITS UN
[10]  B4:→(ZERO≠⍴⎕GETKEY)/B4⋄→ZERO
[11]  B3:SHOWCURSOR⋄→BUTTON/B3⋄→B2
     ∇
```

```
∇HIDECURSOR∇


    ∇ICON ICONNAMS;EXE;H;R2;K;MU;ISIZE;N;E;ICONLOC;RECT;STOP;KW;L;C;OBJ;OPT;M;A
     ⎕ELX←'⎕DM'⋄CLIPRECT 0 0 299 507⋄ERASERECT 0 0 299 507
     ICONNAMS←' ' CHARMAT ICONNAMS⋄STOP←ONE⋄K←ONE⋄R2←0 4⍴IZ⋄ICONLOC←R2⋄H←12
     E←(⍴ICONNAMS)[ONE]⋄EXE←0 0⍴IZ⋄A←A1,A2,A3⋄DELETEMENU 2 3 4
     OPTIONSN SETMENU OPTIONS⋄ONE SETMENU DESKTOP⋄DRAWMENUBAR
     B1:MU←DLBS2 ICONNAMS[K;]⋄N←⍎MU,'ICON'⋄ISIZE←PICTFRAME N
     C←FIVE,H,(FIVE+ISIZE[THREE]-ISIZE[ONE]),H+ISIZE[FOUR]-ISIZE[TWO]
     ICONLOC←ICONLOC,[ONE]1 4⍴C⋄H←H+13+ISIZE[FOUR]-ISIZE[TWO]
     EXE←EXE APPEND ⍎MU,'ICONEXE'⋄C DRAWPICTURE N
     K←K+ONE⋄→(E≥K)/B1⋄E←⍳E⋄DRAWMFIVEICON⋄H←IZ⋄N←'PRIME_'
     B4:M←⎕GETKEY⋄→(THREE≠⍴M)/B4⋄→(TWO=M[ONE])/A0⋄M←M[2 3]⋄DELETEMENU OPTIONSN
     L←(M PTINRECT ICONLOC)/E⋄→(ZERO≠⍴L)/B2
     L←(M PTINRECT R2)/H⋄→(ZERO=⍴L)/B5⋄⍎,KW[L;]
     B5:OPTIONSN SETMENU OPTIONS⋄→B4
     B2:⍎,EXE[L;]⋄H←⍳(⍴R2)[ONE]⋄OPTIONSN SETMENU OPTIONS⋄→STOP/B4
     STANDMENUBAR⋄EXITMSG⋄→ZERO
     A0:→(M[TWO]≠OPTIONSN)/B4⋄→A[M[THREE]]
     A1:L←⎕GETBITS LOADRECT⋄K←'Enter File Name:' FILESAVE N⋄L ⎕PUTBITS LOADRECT
     L←IZ⋄→(ZERO=⍴K)/B4⋄N←(ONE-(⌽K)⍳':')↑K⋄C←⎕ELX⋄⎕ELX←'→B3'
     K←K DATASAVE PRIMEDATA⋄⎕ELX←C⋄ERRORWINDOW 'SAVE COMPLETED'⋄→B4
     B3:ERRORWINDOW 'DISK/WORKSPACE FULL:  SAVE NOT COMPLETE - FILE DELETED'
     ⎕ELX←'→B6'⋄N ⎕FERASE ⌈/⎕FNUMS
     B6:⎕ELX←C⋄→B4
     A2:L←⎕GETBITS LOADRECT⋄K←FILEOPEN 'PRIME_'⋄L ⎕PUTBITS LOADRECT⋄L←IZ
     →(ZERO=⍴K)/B4⋄N←⎕EX PRIMEDATA⋄N←(ONE-(⌽K)⍳':')↑K⋄DATALOAD K
     RECT←0 5⍴IZ⋄R2←0 4⍴IZ⋄H←IZ⋄CLIPRECT 0 0 299 507⋄ERASERECT 70 0 299 507
     DRAWMFIVEICON⋄→B4
     A3:L←⎕GETBITS LOADRECT⋄K←FILEOPEN 'PRIME_'⋄L ⎕PUTBITS LOADRECT
     L←IZ⋄→(ZERO=⍴K)/B4
     L←ONE+⌈/ZERO,⎕FNUMS⋄K ⎕FHTIE L⋄K ⎕FERASE L⋄ERRORWINDOW 'FILE DELETED'⋄→B4
    ∇


∇INITCURSOR∇


∇INVERTCURSOR∇


∇INVERTROUNDRECT∇


    ∇A IR B;SCOL;SROW;C1;C2
     →(ZERO=⍴S)/ZERO⋄SCOL←⌈S+18⋄SROW←S-18×SCOL-ONE
     C1←S2+SEVEN+SIX×A⌈FL[SCOL]-LC⋄C2←ONE+S2+SIX×(B+ONE)⌊FL[SCOL+ONE]-LC
     B1:INVERTRECT (¯9+S1+SROW×ELEVEN),C1,(TWO+S1+SROW×ELEVEN),[1.1]C2
    ∇
```

```
∇IR2;SROW;SCOL;C1;C2;G1
[1]  SROW←⌈((NTWO+M[ONE])-S1)÷ELEVEN◊→(SROW<ONE)/ZERO◊→(SROW>N)/ZERO
[2]  SCOL←+/FL<LC+(M[TWO]-S2+FIVE)÷SIX◊C1←SROW+N×SCOL-ONE◊→(AMAX<C1)/ZERO
[3]  →(C1∊S)/B1◊→((-C1)∊RS)/B4◊→(MAX=⍴S)/B3◊S←S,C1
[4]  B2:C1←S2+SEVEN+SIX×ZERO⌈FL[SCOL]-LC◊C2←ONE+S2+SIX×(RC+ONE)⌊FL[SCOL+ONE]-LC
[5]  INVERTRECT (¯9+S1+SROW×ELEVEN),C1,(TWO+S1+SROW×ELEVEN),C2
[6]  ERASERECT (C+¯9 116),C+0 145
[7]  MOVETO C+0 116◊TEXTFACE ONE◊DRAWTEXT FOUR↑⍕⍴S◊TEXTFACE ZERO◊→ZERO
[8]  B1:S←(S≠C1)/S◊→B2
[9]  B3:G1←'YOU CANNOT SELECT MORE THAN ',(⍕MAX),' ENTR',(3+¯4×1=MAX)↑'IESY'
[10] ERRORWINDOW G1◊→ZERO
[11] B4:ERRORWINDOW MSG3
∇


∇JOBAUDIOGET OBJ;A;VAL
[1]  B4:VAL←'JOBAUDIO'◊A←7 1 GETINPUT NONE↓,RECT[L;]
[2]  →(OPT=2 3)/B1,B2◊JOBAUDIO[OBJ]←A◊→ZERO
[3]  B1:JOBAUDIOLIST[JOBAUDIO[OBJ];] CLIST NONE↓,RECT[L;]◊→ZERO
[4]  B2:JOBAUDIO[OBJ]←JOBAUDIODEF◊→B1
∇


∇JOBDEFGET OBJ;A
[1]  B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]  PERFDATA2[PERFDATA1⍳-JOBNN[OBJ]]←A◊→ZERO
[3]  B1:(JOBTIMEDEF CVTNUM PERFDATA2[PERFDATA1⍳-JOBNN[OBJ]]) CLIST ¯1↓,RECT[L;]
[4]  →ZERO
[5]  B2:PERFDATA2[PERFDATA1⍳-JOBNN[OBJ]]←JOBTIMEDEF◊→B1
[6]  B3:HELPWINDOW XJOBDEFHELP◊→B4
∇


∇JOBINFOGET OBJ;B;MSG;C
[1]  →(TWO≠⎕NC B←'JOB',(⍕JOBNN[OBJ]),'INFO')/B1◊C←⍎B
[2]  B3:MSG←(DLBS2 ,JOBLIST[OBJ+ONE;]),' Job Description'
[3]  C←((TEN,⌈SIX+1.25×⍴MSG)⌈⍴C)↑C
[4]  C←Z1 C◊C←⍉DLBS3 ⍉DLBS3 C◊→(ZERO=×/⍴C)/B2◊⍎B,'←C'◊→ZERO
[5]  B1:C←0 0⍴''◊→B3
[6]  B2:C←⎕EX B
∇


∇JOBMEMORYGET OBJ;A
[1]  B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]  JOBMEMORY[OBJ]←A◊→ZERO
[3]  B1:(JOBMEMORYDEF CVTNUM JOBMEMORY[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]  B2:JOBMEMORY[OBJ]←JOBMEMORYDEF◊→B1
[5]  B3:HELPWINDOW XJOBMEMORYHELP◊→B4
∇


∇JOBMULTIPLEXGET OBJ;A
[1]  B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]  JOBMULTIPLEX[OBJ]←A◊→ZERO
[3]  B1:(JOBMULTIPLEXDEF CVTNUM JOBMULTIPLEX[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
```

```
[4]    B2:JOBMULTIPLEX[OBJ]←JOBMULTIPLEXDEF◊→B1
[5]    B3:HELPWINDOW XJOBMULTIPLEXHELP◊→B4
     ∇


     ∇JOBPERFGET OBJ;G1;G2;G3;G4;C;D;J;E;H;VAL;R;K;F;G5;A;B;MSG2;HELPMSG
[1]    F←JOBNN[OBJ]◊G1←PERFDATA1⍳-F◊G2←NONE+((G1↓PERFDATA1)<ZERO)⍳ONE
[2]    J←G2↑G1↓PERFDATA1◊E←G2↑G1↓PERFDATA2◊HELPMSG←XJOBPERFHELP
[3]    C←E,((⍴CREWLIST)[ONE]-G2+ONE)⍴NTWO◊G3←CREWNN⍳J
[4]    H←CREWLIST[ONE+G3;]⍪CREWLIST ELIM ONE,ONE+G3◊D←⍴C◊G4←⍕(D,ONE)⍴C
[5]    G4[(C<ZERO)/⍳D;]←' '◊H←H,'/',G4,' '◊VAL←'ENTER PERFORMANCE TIME:  '
[6]    H[(E=NONE)/⍳G2;ONE↓⍴H]←'*'◊MSG2←''◊→(ZERO>PERFDATA2[G1])/B1
[7]    MSG2←'Default Performance Time = ',(⍕PERFDATA2[G1]),' Minutes'
[8]    B1:K←'H VAL ','Performance Data For ',DLBS2 JOBLIST[IZ⍴ONE+OBJ;]
[9]    R←C EDA K◊PERFDATA2[G1+⍳G2]←G2↑R◊R←G2↓R◊A←(R≠NTWO)/R
[10]   G5←(R≠NTWO)/(⍳NONE+(⍴CREWLIST)[ONE]) ELIM G3
[11]   PERFDATA1←((G1+G2)↑PERFDATA1),CREWNN[G5],(G1+G2)↓PERFDATA1
[12]   PERFDATA2←((G1+G2)↑PERFDATA2),A,(G1+G2)↓PERFDATA2
     ∇


     ∇JOBPOWERGET OBJ;A
[1]    B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]    JOBPOWER[OBJ]←A◊→ZERO
[3]    B1:(JOBPOWERDEF CVTNUM JOBPOWER[OBJ]) CLIST NONE↓,RECT[L;]◊→0
[4]    B2:JOBPOWER[OBJ]←JOBPOWERDEF◊→B1
[5]    B3:HELPWINDOW XJOBPOWERHELP◊→B4
     ∇


     ∇JOBSIZEGET OBJ;A
[1]    B4:A←4 1 1 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]    JOBSIZE[OBJ]←A◊→ZERO
[3]    B1:(JOBSIZEDEF CVTNUM JOBSIZE[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]    B2:JOBSIZE[OBJ]←JOBSIZEDEF◊→B1
[5]    B3:HELPWINDOW XJOBSIZEHELP◊→B4
     ∇


     ∇JOBTOOLGET OBJ;G1;G2;G3;G4;C;D;J;E;H;VAL;R;K;F;G5;A;B;MSG2;HELPMSG
[1]    F←JOBNN[OBJ]◊G1←TOOLUSE1=F◊C←G1/TOOLUSE2◊HELPMSG←XJOBTOOLHELP
[2]    G2←TOOLUSE1<ZERO◊J←G2/TOOLUSE1◊J←-J[G1/+\G2]◊G3←TOOLNN⍳J
[3]    H←TOOLLIST[ONE+G3;]⍪TOOLLIST ELIM ONE,ONE+G3
[4]    C←C,((⍴H)[ONE]-⍴C)⍴TOOLUSEDEF◊G5←-((NONE+(⍴TOOLLIST)[ONE])↑TOOLNN) ELIM G3
[5]    D←⍴C◊G4←⍕(D,ONE)⍴C◊G4[(C=TOOLUSEDEF)/⍳D;]←' '◊H←H,'/',G4,' '
[6]    VAL←'ENTER NUMBER OF THIS TOOL NEEDED:   '◊K←⍴J◊MSG2←''
[7]    R←C EDA 'H VAL Tools Needed For ',DLBS2 JOBLIST[IZ⍴ONE+OBJ;]
[8]    TOOLUSE2[G1/⍳⍴TOOLUSE2]←K↑R◊R←K↓R◊A←(R≠K↓C)/R◊G5←(R≠K↓C)/G5
[9]    →(ZERO=⍴A)/B2◊K←ONE
[10]   B1:TOOLUSE1←(B↑TOOLUSE1),F,(B←TOOLUSE1⍳G5[K])↓TOOLUSE1
[11]   TOOLUSE2←(B↑TOOLUSE2),A[K],B↓TOOLUSE2◊K←K+ONE◊→(K≤⍴A)/B1
[12]   B2:G1←(TOOLUSE1<ZERO)∨TOOLUSE2>ZERO◊TOOLUSE1←G1/TOOLUSE1
[13]   TOOLUSE2←G1/TOOLUSE2
     ∇


     ∇JOBTRANSGET OBJ;A
[1]    B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]    JOBTRANS[OBJ]←A◊→ZERO
```

```
[3]   B1:(JOBTRANSDEF CVTNUM JOBTRANS[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:JOBTRANS[OBJ]←JOBTRANSDEF◊→B1
[5]   B3:HELPWINDOW XJOBTRANSHELP◊→B4
    ∇


    ⍒LINETO⍒


    ∇A MANASGET OBJ;DEF;B
[1]   DEF←(MANAS[A;OBJ]≠MANASDEF[A])×MANAS[A;OBJ]-MANAS[THREE-A;OBJ]
[2]   DEF←MANAS[THREE-A;OBJ]+DEF
[3]   DEF←(3,DEF,((A=1)/ZERO),(MANAS[3-A;OBJ]≠MANASDEF[3-A])⍴MANAS[THREE-A;OBJ])
[4]   B4:B←DEF GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3◊MANAS[A;OBJ]←B◊→0
[5]   B1:(MANASDEF[A] CVTTIME MANAS[A;OBJ]) CLIST  NONE↓,RECT[L;]◊→ZERO
[6]   B2:MANAS[A;OBJ]←MANASDEF[A]◊→B1
[7]   B3:HELPWINDOW XMANASHELP◊→B4
    ∇


    ∇MANCREWGET OBJ;MAX;LIST;MSG;MSG2;C;H;G2;G1
[1]   H←+\MANCREW<ZERO◊G2←(MANCREW∊-MANNN[OBJ])/H◊C←CREWNN⍳ONE↓(H∊G2)/MANCREW
[2]   LIST←CREWLIST[ONE+C;]⍪CREWLIST ELIM ONE,ONE+C◊MAX←SEVEN
[3]   MSG2←'Please Select The Crewmembers For This Flight Plan'
[4]   G1←C,(⍳NONE+(⍴CREWLIST)[ONE]) ELIM C
[5]   MSG←(DLBS2 ,MANLIST[OBJ+ONE;]),' Crewmember List'◊C←CHOICE ⍳⍴C
[6]   MANCREW←((H⍳G2)↑MANCREW),CREWNN[G1[C]],(NONE+H⍳G2+ONE)↓MANCREW
    ∇


    ∇MANJOBGET OBJ;MAX;LIST;MSG;MSG2;C;H;G2;G1
[1]   H←+\MANJOB<ZERO◊G2←(MANJOB∊-MANNN[OBJ])/H◊C←JOBNN⍳ONE↓(H∊G2)/MANJOB
[2]   LIST←JOBLIST[ONE+C;]⍪JOBLIST ELIM ONE,ONE+C◊MAX←ZERO
[3]   MSG2←'Please Select The Jobs For This Flight Plan'
[4]   G1←C,(⍳NONE+(⍴JOBLIST)[ONE]) ELIM C
[5]   MSG←(DLBS2 ,MANLIST[OBJ+ONE;]),' Job List'◊C←CHOICE ⍳⍴C
[6]   MANJOB←((H⍳G2)↑MANJOB),JOBNN[G1[C]],(NONE+H⍳G2+ONE)↓MANJOB
    ∇


    ∇MANMEMORYGET OBJ;A
[1]   B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]   MANMEMORY[OBJ]←A◊→ZERO
[3]   B1:(MANMEMORYDEF CVTNUM MANMEMORY[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:MANMEMORY[OBJ]←MANMEMORYDEF◊→B1
[5]   B3:HELPWINDOW XMANMEMORYHELP◊→B4
    ∇


    ∇MANMULTIPLEXGET OBJ;A
[1]   B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]   MANMULTIPLEX[OBJ]←A◊→ZERO
[3]   B1:(MANMULTIPLEXDEF CVTNUM MANMULTIPLEX[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:MANMULTIPLEX[OBJ]←MANMULTIPLEXDEF◊→B1
[5]   B3:HELPWINDOW XMANMULTIPLEXHELP◊→B4
    ∇
```

```
∇MANPOWERGET OBJ;A
 B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]⋄→(OPT=2 3 6)/B1,B2,B3
 MANPOWER[OBJ]←A⋄→ZERO
 B1:(MANPOWERDEF CVTNUM MANPOWER[OBJ]) CLIST NONE↓,RECT[L;]⋄→ZERO
 B2:MANPOWER[OBJ]←MANPOWERDEF⋄→B1
 B3:HELPWINDOW XMANPOWERHELP⋄→B4
∇
```

```
∇MANSCHED OBJ;C;R;F;J;JN;CN;D;K;GG;A;B;E;G;W;PD;CL;JL;M;G1;G2;H;GT
 F←'Any Changes You Have Made To The Active Workspace Will Not Be '
 →(~YESNOWINDOW F,'Saved.  Do You Still Wish To Proceed?')/ZERO
 F←ONE+⌈/ZERO,⎕FNUMS⋄W←WS⋄((W,SCHEDDISK,'MAN'),⍕MANNN[OBJ]) FILEHCREATE F
 A←⎕TCNL⋄E←A,'N',A⋄G←A,'C',A⋄J←(MANJOB⍳-MANNN[OBJ])↓MANJOB⋄→(ZERO≠J)/B10
 ERRORWINDOW 'There Are No Jobs Assigned To This Flight Plan'⋄→B3
 B10:J←(NONE+(J<ZERO)⍳ONE)↑J⋄R←JOBNN⍳J⋄JL←⍴J⋄JN←JOBLIST[ONE+R;]⋄K←A,(⍕JL),E
 ('JOBNAMES',A,(⍕⍴JN),G,(,JN),A) ⎕FAPPEND F⋄C←(MANCREW⍳-MANNN[OBJ])↓MANCREW
 C←(NONE+(C<ZERO)⍳ONE)↑C⋄CN←CREWLIST[ONE+CREWNN⍳C;]⋄CL←⍴C
 ('CREWNAMES',A,(⍕⍴CN),G,(,CN),A) ⎕FAPPEND F
 D←MANPOWER[OBJ],MANTRANS[OBJ],MANMEMORY[OBJ],MANMULTIPLEX[OBJ]
 ('RESLIM',A,'4',E,(⍕D),A) ⎕FAPPEND F⋄GG←DLBS2 ,MANLIST[ONE+OBJ;]
 ('NAME',A,(⍕⍴GG),G,GG,A) ⎕FAPPEND F
 H←MANAS[;IZ⍴OBJ]⋄→(∧/H≠MANASDEF)/B6⋄M←'Both The Start And End Dates Must '
 ERRORWINDOW M,'Be Defined In Order To Produce A Schedule.'⋄→B3
 B6:('STARTEND',A,'2',E,(⍕H),A) ⎕FAPPEND F
 H←JOBSIZE[R]⋄('JOBSIZE',K,(⍕H),A) ⎕FAPPEND F
 ('JOBAUDIO',K,(⍕JOBAUDIO[R]),A) ⎕FAPPEND F
 GG←⍉(FOUR,⍴J)⍴JOBPOWER[R],JOBTRANS[R],JOBMEMORY[R],JOBMULTIPLEX[R]
 G1←⌈⌿GG⋄G2←(D<ZERO)∨G1≤D⋄→(∧/G2)/B4⋄G2←G2⍳ZERO⋄G1←GG[;G2]⍳G1[G2]
 M←(DLBS RESNAME[G2;]),' Usage Exceeded For Job '
 ERRORWINDOW M,(⍕J[G1]),': ',(DLBS2 JN[G1;]),'.'⋄→B3
 B4:('JOBRES',A,(⍕JL,FOUR),E,(⍕,GG),A) ⎕FAPPEND F
 M←PERFDATA1⍳-J⋄GT←PERFDATA2[M]⋄('JOBTDEF',K,(⍕GT),A) ⎕FAPPEND F
 PD←⍉((ONE⌈CL),JL)⍴GT⋄R←+\PERFDATA1<ZERO⋄D←ONE⋄G1←(R⍳R[M]+ONE)-M+ONE⋄GG←IZ
 B1:→(ZERO=H[D])/B9⋄B←M[D]+⍳G1[D]⋄G2←PERFDATA1[B]⋄R←G2∊C⋄G2←R/C⍳G2
 PD[D;G2]←R/PERFDATA2[B]⋄R←PD[D;]⋄G2←(R≠JOBTIMEDEF)/R⋄→(H[D]≤⍴G2)/B2
 M←'There Are Not Sufficient Crewmembers Available For Job '
 ERRORWINDOW M,(⍕J[D]),': ',(DLBS2 JN[D;]),'.'
 B3:((W,SCHEDDISK,'MAN'),⍕MANNN[OBJ]) ⎕FERASE F⋄→ZERO
 B9:→(GT[D]=JOBTIMEDEF)/B8⋄GG←GG,GT[D]⋄→B7
 B8:M←'There Is No Default Job Time For Job ',(⍕J[D]),': ',DLBS2 JN[D;]
 ERRORWINDOW M,', Which Requires No Crewmembers.'⋄→B3
 B2:GG←GG,G2[(⍋G2)[H[D]]]
 B7:D←D+ONE⋄→(D≤JL)/B1
 →((⌈/GG)≤--/MANAS[;IZ⍴OBJ])/B5
 M←'The Minimum Completion Time For Job ',(⍕J[GG⍳⌈/GG]),': '
 M←M,(DLBS2 JN[J[GG⍳⌈/GG];]),' Is Longer Than The Flight Plan Duration.'
 ERRORWINDOW M⋄→B3
 B5:('PD',A,(⍕JL,ONE⌈CL),E,(⍕,PD),A) ⎕FAPPEND F
 ('MPD',K,(⍕GG),A) ⎕FAPPEND F⋄⎕FUNTIE F
 (W,SCHEDDISK,'MAN') FILEHCREATE F
 ((W,SCHEDDISK,'MAN',⍕MANNN[OBJ]),A) ⎕FAPPEND F
 ⎕FUNTIE F⋄ERASERECT 0 0 299 507⋄⎕QLOAD W,SCHEDDISK,SCHEDWS
∇
```

```
∇MANTRANSGET OBJ;A
 B4:A←4 1 0 0 GETINPUT NONE↓,RECT[L;]⋄→(OPT=2 3 6)/B1,B2,B3
 MANTRANS[OBJ]←A⋄→ZERO
 B1:(MANTRANSDEF CVTNUM MANTRANS[OBJ]) CLIST NONE↓,RECT[L;]⋄→ZERO
 B2:MANTRANS[OBJ]←MANTRANSDEF⋄→B1
 B3:HELPWINDOW XMANTRANSHELP⋄→B4
```

```
∇
```

```
∇MOVETO∇

∇PAINTRECT∇

∇PENMODE∇

∇PENNORMAL∇

∇PENPAT∇

∇PENSIZE∇

∇PICTFRAME∇
```

```
∇R←PRIMEDATA;G;E;F;H;J
 R←⎕NL TWO⋄G←(R[;ONE]='C')⌿R⋄G←(G[;TWO]='R')⌿G⋄E←(R[;TWO]='O')⌿R
 H←(E[;ONE]='J')⌿E⋄E←(E[;ONE]='T')⌿E⋄F←(R[;ONE]='M')⌿R⋄F←(F[;TWO]='A')⌿F
 J←(R[;ONE]='S')⌿R⋄J←(J[;TWO]='T')⌿J
 R←G,[ONE]H⍪F⍪E⍪J⍪(TWO,ONE↓⍴E)↑2 9⍴'PERFDATA1PERFDATA2'
∇
```

```
∇PTINRECT∇
```

```
∇R←B REPLYWINDOW A;OS;K;M;D;UN;UN2;P;KK
 D←22 20 70 480⋄UN←22 20 70 484⋄UN2←⎕GETBITS UN⋄ERASERECT D⋄KK←ONE
 FRAMERECT 2 4⍴D,24 22 68 478
 TEXTSIZE NINE⋄TEXTFACE ONE⋄K←35+FIVE×FOUR-THREE⌊⌈(⍴,A)÷45
 B1:MOVETO K,50⋄OS←45-(⌽45↑A)⍳' '⋄P←OS<ZERO⋄OS←OS+45×P⋄→((KK=3)∧OS<⍴A)/B5
 DRAWTEXT DLBS2 OS↑A⋄KK←KK+ONE⋄A←(OS+~P)↓A⋄K←K+ELEVEN⋄→(ZERO≠⍴A)/B1
 B6:TEXTFACE ZERO
 R←B GETINPUT (¯7 3+GETPEN),(ONE↑GETPEN),477⋄UN2 ⎕PUTBITS UN
 B4:→(ZERO≠⍴⎕GETKEY)/B4⋄→ZERO
 B5:DRAWTEXT (41↑A),'...'⋄→B6
∇
```

```
∇A REVISENAME G;H;D;R;E;G1;P;B;J
 B←ONE↓A
 B1:R←ONE GETINPUT G⋄→(OPT=2 3 6)/B2,B3,B4⋄D←⍎B,'LIST'
 H←'Do You Wish To Replace ',(DLBS2 ,D[OBJ+ONE;]),' By ',R,'?'
 H←~YESNOWINDOW H⋄→H/B2
 ⍎B,'LIST←D[⍳OBJ;] APPEND R APPEND ((ONE+OBJ),ZERO)↓D'
 ((ONE,⍴R)⍴R) CLIST G⋄→ZERO
 B2:(⍎B,'LIST')[,OBJ+ONE;] CLIST G⋄→ZERO
 B3:H←'Are You Sure You Wish To Delete ',DLBS2 ,(D←⍎B,'LIST')[OBJ+ONE;]
 H←~YESNOWINDOW H,' From The System?'⋄→H/B2⋄G1←⍎B,'NN[OBJ]'
 E←(⍴D)[ONE]⍴ONE⋄E[OBJ+ONE]←ZERO⋄→(B5,B6,B7,B8,B10)[A FIND3 COMLIST]
 B5:CREWLIST←E⌿CREWLIST⋄E←ONE↓E
 CREWMS←E/CREWMS⋄CREWAS←E/CREWAS⋄CREWN←(G1≠(⍴E)↓CREWNN)⌿CREWN
```

```
[13]  CREWBD←E/CREWBD⋄H←~PERFDATA1=G1⋄PERFDATA1←H/PERFDATA1
[14]  PERFDATA2←H/PERFDATA2⋄MANCREW←(~MANCREW=G1)/MANCREW⋄CREWRANK←E/CREWRANK
[15]  CREWHT←E/CREWHT⋄TOOLUSERS←TOOLUSERS×TOOLUSERS≠G1
[16]  B9:⍎B,'NN←(G1≠',B,'NN)/',B,'NN'⋄E←⎕EX B,(⍕G1),'INFO'⋄'' FORMGET A⋄→ZERO
[17]  B6:JOBLIST←E⌿JOBLIST⋄E←ONE↓E⋄JOBMEMORY←E/JOBMEMORY⋄JOBPOWER←E/JOBPOWER
[18]  JOBN←(G1≠(ρE)↓JOBNN)⌿JOBN
[19]  JOBAUDIO←E/JOBAUDIO⋄JOBMULTIPLEX←E/JOBMULTIPLEX⋄JOBTRANS←E/JOBTRANS
[20]  H←+\PERFDATA1<ZERO⋄H←~H∊(PERFDATA1=-G1)/H⋄PERFDATA2←H/PERFDATA2
[21]  PERFDATA1←H/PERFDATA1⋄H←~TOOLUSE1=G1⋄TOOLUSE1←H/TOOLUSE1
[22]  TOOLUSE2←H/TOOLUSE2⋄JOBSIZE←E/JOBSIZE⋄→B9
[23]  B7:MANLIST←E⌿MANLIST⋄E←ONE↓E⋄MANMEMORY←E/MANMEMORY⋄MANTRANS←E/MANTRANS
[24]  MANN←(G1≠(ρE)↓MANNN)⌿MANN
[25]  MANPOWER←E/MANPOWER⋄MANMULTIPLEX←E/MANMULTIPLEX⋄MANAS←E/MANAS
[26]  H←+\MANCREW<ZERO⋄MANCREW←(~H∊(MANCREW∊-G1)/H)/MANCREW⋄H←+\MANJOB<ZERO
[27]  MANJOB←(~H∊(MANJOB∊-G1)/H)/MANJOB⋄→B9
[28]  B8:TOOLLIST←E⌿TOOLLIST⋄E←ONE↓E⋄TOOLN←(G1≠(ρE)↓TOOLNN)⌿TOOLN
[29]  H←+\TOOLUSE1<ZERO⋄H←~H∊(TOOLUSE1∊-G1)/H⋄TOOLUSE2←H/TOOLUSE2
[30]  TOOLUSE1←H/TOOLUSE1⋄TOOLVOL←E/TOOLVOL
[31]  H←+\TOOLSTOW<ZERO⋄TOOLSTOW←(~H∊(TOOLSTOW∊-G1)/H)/TOOLSTOW
[32]  H←+\TOOLSTAT<ZERO⋄TOOLSTAT←(~H∊(TOOLSTAT∊-G1)/H)/TOOLSTAT
[33]  H←+\TOOLHIST<ZERO⋄TOOLHIST←(~H∊(TOOLHIST∊-G1)/H)/TOOLHIST
[34]  H←+\TOOLUSEX<ZERO⋄TOOLUSEX←(~H∊(TOOLUSEX∊-G1)/H)/TOOLUSEX
[35]  H←+\TOOLUSERS<ZERO⋄TOOLUSERS←(~H∊(TOOLUSERS∊-G1)/H)/TOOLUSERS
[36]  H←+\TOOLUSEMODS<ZERO⋄TOOLUSEMODS←(~H∊(TOOLUSEMODS∊-G1)/H)/TOOLUSEMODS⋄→B9
[37]  B10:STOWLIST←E⌿STOWLIST⋄E←ONE↓E⋄STOWN←(G1≠(ρE)↓STOWNN)⌿STOWN
[38]  STOWVOL←E/STOWVOL⋄STOWX←E/STOWX⋄STOWMOD←E/STOWMOD⋄STOWTHETA←E/STOWTHETA
[39]  H←TOOLSTOW=G1⋄J←+\TOOLSTOW<ZERO⋄J←TOOLUSE1⍳TOOLSTOW[J⍳H/J]
[40]  TOOLUSE2[J]←TOOLUSE2[J]-ONE⋄TOOLSTOW←(~H)/TOOLSTOW⋄TOOLSTAT←(~H)/TOOLSTAT
[41]  TOOLHIST←(TOOLHIST≠G1)/TOOLHIST⋄→B9
[42]  B4:HELPWINDOW REVISENAMEHELP⋄→B1
    ∇


    ∇B REVISENICKNAME OBJ;G1;G2;G3;JLIST
[1]   B←ONE↓B⋄G1←⍎B,'N'⋄G2←⍎B,'NN'⋄G3←(-ONE↑ρG1)↑G2⋄JLIST←(G3=G2[OBJ])⌿G1
[2]   ED 'JLIST Nicknames For ',DLBS2 ,⍎B,'LIST[OBJ+ONE;]'
[3]   ⍎B,'NN←((-ρG3)↓G2),((G3≠G2[OBJ])/G3),(ρJLIST)[ONE]ρG2[OBJ]'
[4]   ⍎B,'N←DLBS3 ((G3≠G2[OBJ])⌿G1) APPEND JLIST'
    ∇


    ∇S
[1]   ⎕SA←ACTIONSTOP⋄T⋄CLEAR⋄ICON 'XCREW XJOB XMAN XTOOL XSTOW APL'
    ∇


    ∇SCHED
[1]   ⎕QLOAD WS,SCHEDDISK,SCHEDWS
    ∇


   ⍫SCROLLRECT⍫


    ∇SEARCHCTL OBJ;F;POS;Q;INDEX;DEFBINS;INDEX;USEX;USEMODS;USERS;K;RANKS;R;P;L
[1]   F←-TOOLNN[OBJ]⋄POS←TOOLSTOW⍳F⋄R←TOOLUSE2[TOOLUSE1⍳F]
[2]   INDEX←(POSρZERO),(RρONE),((ρTOOLSTOW)-(POS+R))ρZERO
[3]   DEFBINS←INDEX/TOOLSTOW⋄USEMODS←INDEX/TOOLUSEMODS⋄USEX←INDEX/TOOLUSEX
[4]   USERS←INDEX/TOOLUSERS⋄K←MODMATRIX[USEMODS;FOUR]≠THREE⋄USERS←K/USERS
[5]   USEMODS←K/USEMODS⋄USEX←K/USEX⋄Q←((ρSTOWVOL)-ρSTOWNN)↓STOWNN
```

```
[6]   L←(~Q∊DEFBINS)/⍳⍴Q⋄BINS←Q[L]⋄SEARCHSIZERANK
[7]   USEX SEARCHPROXRANK USEMODS⋄SEARCHJOBRANK⋄SEARCHHISTRANK TOOLHIST
[8]   P←⎕PP⋄ST←INDEX/TOOLSTAT⋄H1←(BINH1 CAPPEND STOWLIST[ONE+Q⍳DEFBINS;]),AV125
[9]   H1←H1,(BINH2 CAPPEND STATLIST[ST;]),AV125⋄H1[TWO;]←'='⋄H1←H1⍪' '
[10]  BINS←BINS[⍒RANKS]⋄K←Q⍳BINS⋄H2←(BINH3 CAPPEND STOWLIST[ONE+K;]),AV125
[11]  G1←NONE SEARCHCVTNUM STOWMOD[K]⋄H2←H2,(BINH4 CAPPEND G1),AV125
[12]  G1←STOWXYZDEF SEARCHCVTNUM STOWX[K]⋄H2←H2,(BINH5 CAPPEND G1),AV125
[13]  G1←STOWXYZDEF SEARCHCVTNUM STOWTHETA[K]⋄H2←H2,BINH6 CAPPEND G1
[14]  H2[TWO;]←'='⋄H2←H2⍪' '⋄G1←CREWLIST[ONE+CREWNN⍳(USERS∊CREWNN)/USERS;]
[15]  G1←G1,NONE CVTNUM (USERS∊CREWNN)/USEMODS⋄H3←BINH7 CAPPEND G1
[16]  H3[THREE;]←'='⋄H3←H3⍪' '⋄H2←H2 CAPPEND H3⋄H1←H1 CAPPEND H2⋄VAL←''
[17]  MSG←'Search Results'⋄MSG2←'Was The Tool Found?'
[18]  MSG3←'Please Click YES or NO'⋄K←ZERO EDB 'H1 VAL ',MSG⋄→(K=ONE)/B1
[19]  K←TOOLSTAT[POS+⍳R]⋄K←(K=SIX)/⍳⍴K⋄→(ZERO=⍴K)/B2⋄TOOLSTAT[POS+K]←TWO⋄→B2
[20]  B1:H1←1 0↓STOWLIST⋄G1←R SEARCHDISPLAY H1⋄→(ZERO=⍴G1)/B2⋄G1←Q[G1]
[21]  SEARCHINFER G1
[22]  B2:⎕PP←P
    ∇


    ∇R←DEF SEARCHCVTNUM NUMS;C
[1]   R←⍕((⍴,NUMS),ONE)⍴NUMS⋄C←(NUMS∊DEF)/⍳⍴,NUMS⋄→(ZERO=⍴C)/ZERO
[2]   R←(0 9⌈⍴R)↑R⋄R[C;]←' '⋄R[C;⍳9]←((⍴C),9)⍴'*NO INFO*'
    ∇


    ∇R←MAX SEARCHDISPLAY LIST;MSG;MSG2
[1]   MSG←'Stowage Locations'⋄MSG2←'Select Bin(s) In Which Tool Was Found'
[2]   R←CHOICE IZ
    ∇


    ∇SEARCHHISTRANK TOOLHIST;MAX;A
[1]   MAX←TEN⋄A←(TOOLHIST⍳F)↓TOOLHIST⋄A←(NONE+(A<ZERO)⍳ONE)↑A
[2]   A←+/BINS∘.=A⋄RANKS←RANKS+A×MAX
    ∇


    ∇SEARCHINFER V;A;B;C;D
[1]   C←TOOLHIST⍳F⋄A←C↓TOOLHIST⋄A←((NONE+(A<ZERO)⍳ONE)↑A),V⋄B←⍴A
[2]   →(B≤R×THREE)/B1⋄A←(B-THREE×R)↓A
[3]   B1:TOOLHIST←(C↑TOOLHIST),A,(C+(B-⍴V)-⍴TOOLHIST)↑TOOLHIST
[4]   D←TOOLSTAT[POS+⍳R]⋄→(ZERO=+/D=TWO)/ZERO⋄B←DEFBINS∊V⋄→(ZERO=+/B)/B2
[5]   C←(B∧D=TWO)/⍳⍴D⋄TOOLSTAT[POS+C]←SIX
[6]   B2:B←+/BINS∘.=A⋄C←+/DEFBINS∘.=A⋄B←(THREE≤B-⌈/C)/BINS⋄→(ZERO=⍴B)/ZERO
[7]   D←TOOLSTAT[POS+⍳R]⋄→(ZERO=+/TWO=D)/ZERO⋄C←(D=TWO)/⍳⍴D⋄A←(⍴C)↑B
[8]   A←(ZERO≠A)/A⋄C←(⍴A)↑C⋄TOOLSTOW[POS+C]←A⋄B←A∊V⋄→(ZERO=+/B)/ZERO
[9]   TOOLSTAT[POS+C,B]←SIX
    ∇


    ∇SEARCHJOBRANK;MAX;M;A;N;B;JOBS;J;P
[1]   MAX←TEN⋄M←TOOLUSE1⍳F⋄N←(ZERO>M↓TOOLUSE1)⍳ONE
[2]   B←(M⍴ZERO),((N-ONE)⍴ONE),((⍴TOOLUSE1)-(N+M-ONE))⍴ZERO
[3]   JOBS←B/TOOLUSE1⋄JOBS←(JOBS≠TOOLUSEDEF)/JOBS⋄J←(~B)/TOOLUSE1⋄J←(J≠F)/J⋄P←IZ
[4]   L1:J←ONE⌽J⋄B←(J<ZERO)⍳ONE⋄N←(B-ONE)↑J⋄P←P,+/N∊JOBS⋄J←(B-ONE)⌽J
[5]   →(J[ONE]≠-TOOLNN[ONE])/L1⋄B←(P>ZERO)/-(TOOLNN≠-F)/TOOLNN⋄P←(P>ZERO)/P
[6]   J←TOOLUSE2[TOOLUSE1⍳B]⋄A←ONE⋄M←(⍴RANKS)⍴ZERO⋄→(ZERO=⍴B)/L3
[7]   L2:N←TOOLSTOW[(TOOLSTOW⍳B[A])+⍳J[A]]⋄M←M+(BINS∊N)×P[A]⋄A←A+ONE⋄→(A≤⍴B)/L2
```

```
[8]   L3:RANKS←RANKS+M×MAX
     ∇


     ∇USEX SEARCHPROXRANK USEMODS;MIN;MAX;A;V;X;D;MODS;DM;B;C;K
[1]   MAX←TEN⋄MIN←FIVE⋄A←(MODMATRIX[STOWMOD;FOUR]=THREE)/⍳⍴STOWMOD⋄V←~L∊A
[2]   L←V/L⋄RANKS←V/RANKS⋄BINS←V/BINS⋄X←STOWX[L]⋄MODS←STOWMOD[L]
[3]   D←(Q∊DEFBINS)/STOWX⋄DM←(Q∊DEFBINS)/STOWMOD
[4]   B←MODS∊DM⋄C←MODS∊USEMODS⋄RANKS←RANKS+MAX×B+C⋄V←X≠STOWXYZDEF
[5]   K←D≠STOWXYZDEF⋄D←K/D⋄B←B×∨/(MODS∘.=K/DM)∧(MODLENGTH+THREE)≥|X∘.-D
[6]   K←USEX≠STOWXYZDEF⋄USEX←K/USEX
[7]   C←C×∨/(MODS∘.=K/USEMODS)∧(MODLENGTH+THREE)≥|X∘.-USEX
[8]   RANKS←RANKS+(B+C)×MIN×V⋄A←∨/∨/[TWO]SEARCHCMATRIX[MODS;DM;USEMODS]
[9]   RANKS←RANKS+A×MIN
     ∇


     ∇SEARCHSIZERANK;A;MAX;VOLS
[1]   MAX←TEN⋄A←STOWVOL[L]≥TOOLVOL[OBJ]
[2]   L←A/L⋄BINS←A/BINS⋄VOLS←STOWVOL[L]
[3]   RANKS←MAX×VOLS∊STOWVOL[Q⍳DEFBINS]
     ∇


     ∇SETMENU∇


     ∇SHOWCURSOR∇


     ∇STANDMENUBAR∇


     ∇STOWMODGET OBJ;A;VAL;B;STOWMODLIST;STOWMODN;STOWMODNN
[1]   B←(⍴MODMATRIX)[ONE]⋄STOWMODNN←⍳B
[2]   STOWMODLIST←((B,SEVEN)⍴'MODULE '),(⍕(B,ONE)⍴STOWMODNN),(B,TWO)⍴': '
[3]   STOWMODLIST←STOWMODLIST,MODNAMES[ONE+MODMATRIX[;FOUR];]⋄STOWMODN←0 0⍴''
[4]   B4:VAL←'STOWMOD'⋄A←7 1 GETINPUT NONE↓,RECT[L;]⋄→(OPT=2 3)/B1,B2
[5]   STOWMOD[OBJ]←A⋄→ZERO
[6]   B1:→(STOWMOD[OBJ]=STOWMODDEF)/B5
[7]   STOWMODLIST[STOWMOD[OBJ];] CLIST NONE↓,RECT[L;]⋄→ZERO
[8]   B2:STOWMOD[OBJ]←STOWMODDEF
[9]   B5:(STOWMODDEF CVTNUM STOWMODDEF) CLIST NONE↓,RECT[L;]
     ∇


     ∇STOWTHETAGET OBJ;A
[1]   B4:A←4 1 0 0 360 GETINPUT NONE↓,RECT[L;]⋄→(OPT=2 3 6)/B1,B2,B3
[2]   STOWTHETA[OBJ]←A⋄→ZERO
[3]   B1:(STOWXYZDEF CVTNUM STOWTHETA[OBJ]) CLIST NONE↓,RECT[L;]⋄→ZERO
[4]   B2:STOWTHETA[OBJ]←STOWXYZDEF⋄→B1
[5]   B3:HELPWINDOW XSTOWTHETAHELP⋄→B4
     ∇


     ∇STOWTOOLGET OBJ;MSG;MSG2;C;H;VAL;MSG3
[1]   H←+\TOOLSTOW<ZERO⋄C←TOOLNN⍳-TOOLSTOW[H⍳(TOOLSTOW=STOWNN[OBJ])/H]
[2]   H←(H1STOW CAPPEND TOOLLIST[ONE+C;]),AV125
[3]   H←H,H3TOOL CAPPEND STATLIST[STATNN⍳(TOOLSTOW=STOWNN[OBJ])/TOOLSTAT;]
```

```
[4]   H[TWO;]←'='◊VAL←''
[5]   MSG←(DLBS2 ,STOWLIST[OBJ+ONE;]),' Contents List'
[6]   MSG2←'Modify This Info Via the Tool Input Form'
[7]   MSG3←'Please Use Tool Info Form To Modify This Data'
[8]   C←ZERO EDA 'H VAL ',MSG
    ∇


    ∇STOWVOLGET OBJ;A;B
[1]   A←+\TOOLSTOW<ZERO◊B←A⍳(TOOLSTOW=STOWNN[OBJ])/A◊B←TOOLNN⍳-TOOLSTOW[B]
[2]   B←⌈/ZERO,TOOLVOL[B]
[3]   B4:A←(4 1 0,B) GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[4]   STOWVOL[OBJ]←A◊→ZERO
[5]   B1:(NONE CVTNUM STOWVOL[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[6]   B2:STOWVOL[OBJ]←STOWVOLDEF◊→B1
[7]   B3:HELPWINDOW XSTOWVOLHELP◊→B4
    ∇


    ∇STOWXGET OBJ;A
[1]   B4:A←(4 1 0 0,MODLENGTH) GETINPUT NONE↓,RECT[L;]◊→(OPT=2 3 6)/B1,B2,B3
[2]   STOWX[OBJ]←A◊→ZERO
[3]   B1:(STOWXYZDEF CVTNUM STOWX[OBJ]) CLIST NONE↓,RECT[L;]◊→ZERO
[4]   B2:STOWX[OBJ]←STOWXYZDEF◊→B1
[5]   B3:HELPWINDOW XSTOWXHELP◊→B4
    ∇


    ∇T
[1]   TEXTNORMAL◊INITCURSOR◊PENNORMAL◊STANDMENUBAR◊⎕FUNTIE ⎕FNUMS
    ∇


    ∇TEXTFACE∇


    ∇TEXTFONT∇


    ∇TEXTNORMAL
[1]   TEXTFONT FOUR◊TEXTSIZE NINE◊TEXTFACE ZERO
    ∇


    ∇TEXTSIZE∇


    ∇TEXTWIDTH∇


    ∇TOOLCONT;WT;SIZE;ULC;TC;LC;RC;BC;MS;D;S1;S2;S3;S4;G1;G2;G3;G4;G5;CR;MSG
[1]   MSG←' ',(DLBS2 ,TOOLLIST[ONE+OBJ;]),' '◊QTY←TOOLUSE2[TOOLUSE1⍳F]
[2]   STOW←QTY↑(TOOLSTOW⍳F)↓TOOLSTOW◊STAT←QTY↑(TOOLSTAT⍳F)↓TOOLSTAT
[3]   TR1←(H1TOOL CAPPEND ((QTY,FIVE)⍴'COPY '),⍕(QTY,ONE)⍴⍳QTY),AV125
[4]   TR2←(H2TOOL CAPPEND STOWLIST[ONE+STOWNN⍳STOW;]),AV125
[5]   TR3←H3TOOL CAPPEND STATLIST[STATNN⍳STAT;]◊WT←TR1,TR2,TR3◊WT[TWO;]←'='
[6]   WT←WT APPEND 'CREATE NEW COPY'
[7]   P←⍳TEN◊L←ZERO◊A←L1,L2,L3,L4,L5,L6,L7,L8,L9,L10◊LC←ZERO◊TC←LC
[8]   TEXTSIZE 12◊TEXTFONT ZERO◊TEXTFACE ZERO
```

```
[9]  SIZE←36 36⌈175 425⌊11 6×1 15+⍴WT◊ULC←100 50◊MS←TEXTWIDTH MSG
[10]  B4:S1←ULC[ONE]◊S2←ULC[TWO]◊S3←S1+SIZE[ONE]◊S4←S2+SIZE[TWO]
[11]  D←¯9 7 1 7+S1,S2,S1,S2◊BC←⌊(¯3+SIZE[ONE])+11◊RC←⌊(¯9+SIZE[TWO])+SIX
[12]  RECT←S1,S2,S3,S4◊ER←1 1 ¯1 ¯1+RECT◊B←¯18 0 18 18+RECT◊UN←TWO↑B◊K←⎕EX 'UN2'
[13]  UN←UN,UN+16×⌈((TWO↓B)-UN)+16◊UN2←⎕GETBITS UN◊TR←¯18 0 1 18+S1,S2,S1,S4
[14]  ERASERECT B◊CR←¯14 10 ¯3 22+S1,S2,S1,S2◊G1←0 ¯1 18 18+S1,S4,S1,S4
[15]  G1 DRAWPICTURE UPARROW◊G2←¯18 ¯1 0 18+S3,S4,S3,S4◊G2 DRAWPICTURE DOWNARROW
[16]  G3←¯1 0 18 18+S3,S2,S3,S2◊G3 DRAWPICTURE LEFTARROW
[17]  G4←¯1 ¯18 18 0+S3,S4,S3,S4◊G4 DRAWPICTURE RIGHTARROW
[18]  G5←¯1 ¯1 18 18+S3,S4,S3,S4◊G5 DRAWPICTURE CORNER
[19]  K←(S1-15)+TWO×⍳SIX◊K←⍉4 6⍴(K-ONE),(SIX⍴TWO+S2),K,SIX⍴S4+16
[20]  RECT←RECT⍪9 4⍴CR,TR,G1,G2,G3,G4,G5◊FRAMERECT K⍪B⍪¯1 0↓RECT
[21]  G1←⌈PFIVE×15+S4+S2-MS⌊S4-S2+30◊K←(S1-17),G1,(S1-ONE),S2+18+S4-G1
[22]  ERASERECT K⍪3 4⍴(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
[23]  MOVETO (S1-THREE),G1◊TEXTSIZE 12◊TEXTFONT ZERO◊MSG DRAWMSG MS,MS⌊S4-S2+30
[24]  TEXTNORMAL◊S5←0 18 17 ¯18+S3,S2,S3,S4◊S6←18 0 ¯18 17+S1,S4,S3,S4
[25]  B14:MOVETO 10 8+S1,S2◊DRAWTEXT (BC,RC)↑(TC,LC)↓WT
[26]  →(∨/L=6 7 10)/B18◊GRAYPAT FILLRECT S6◊G1←(BC×¯36+S3-S1)+TWO×(⍴WT)[ONE]
[27]  G2←S1+18+((TC+PFIVE×BC)+(⍴WT)[ONE])×(S3-S1)-36
[28]  S6←(⌊G2-G1),S4,((S3-18)⌊⌊G2+G1),17+S4◊ERASERECT S6◊FRAMERECT S6
[29]  RECT[NINE;]←S6◊→(∨/L=4 5 9)/B2
[30]  B18:GRAYPAT FILLRECT S5◊G5←(RC×¯36+S4-S2)+TWO×(⍴WT)[TWO]
[31]  G2←S2+18+((LC+PFIVE×RC)+(⍴WT)[TWO])×S4-S2+36
[32]  S5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◊ERASERECT S5◊FRAMERECT S5
[33]  RECT[TEN;]←S5
[34]  B2:M←⎕GETKEY◊→(THREE≠⍴M)/B2◊→(TWO=M[ONE])/B2◊M←M[2 3]
[35]  L←ONE↑(M PTINRECT RECT)/P◊→(ZERO=L)/B2◊→A[L]
[36]  B3:⎕SOUND BEEP◊→B2
[37]  L1:→BUTTON/L1◊K←TOOLW◊→K/B14◊→B2
[38]  L2:→BUTTON/L2◊UN2 ⎕PUTBITS UN◊→ZERO
[39]  L3:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
[40]  B7:→(~BUTTON)/B5◊FRAMERECT M1◊M1←B+4⍴(0 0⌈GETMOUSE)-M◊FRAMERECT M1◊→B7
[41]  B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
[42]  B30:FRAMERECT M1◊PENPAT BLACKPAT◊UN2 ⎕PUTBITS UN◊PENMODE EIGHT◊→B4
[43]  L4:→(TC≤ZERO)/B3◊TC←TC-ONE◊ERASERECT ER◊→B14
[44]  L5:→(TC≥(⍴WT)[ONE]-BC-TWO)/B3◊TC←TC+ONE◊ERASERECT ER◊→B14
[45]  L6:→(LC≤ZERO)/B3◊HS←(⌽(LC-ONE)↑WT[ONE;])⍳AV125◊LC←LC-LC⌊HS
[46]  ERASERECT ER◊→B14
[47]  L7:G3←(⍴WT)[TWO]-RC◊→(LC≥G3)/B3◊HS←((RC+LC+ONE)↓WT[ONE;])⍳AV125
[48]  LC←LC+HS⌊G3◊ERASERECT ER◊→B14
[49]  L8:M1←B◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[50]  B8:→(~BUTTON)/B9◊FRAMERECT M1◊M1←B[1 2],(18 18+ULC)+36 36⌈⌈SIZE+GETMOUSE-M
[51]  FRAMERECT M1◊→B8
[52]  B9:SIZE←36 36⌈SIZE+GETMOUSE-M◊→B30
[53]  L9:M1←S6◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[54]  B21:→(~BUTTON)/B22◊FRAMERECT M1
[55]  K←(18+S1-S6[1])⌈(S3-18+S6[3])⌊1↑GETMOUSE-M◊M1←S6+4⍴K,0◊FRAMERECT M1◊→B21
[56]  B22:FRAMERECT M1◊TC←TC+⌊PFIVE+K×(⍴WT)[ONE]+S3-S1+36
[57]  B31:ERASERECT ER◊PENPAT BLACKPAT◊PENMODE EIGHT◊→B14
[58]  L10:M1←S5◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[59]  B23:→(~BUTTON)/B24◊FRAMERECT M1
[60]  K←(18+S2-S5[2])⌈(S4-18+S5[4])⌊1↓GETMOUSE-M◊M1←S5+4⍴0,K◊FRAMERECT M1◊→B23
[61]  B24:FRAMERECT M1◊LC←LC+⌊PFIVE+K×(⍴WT)[TWO]+S4-S2+36◊→B31
     ∇


     ∇TOOLCONT1;HS;TR;B;RECT;K;KK;WT1;S5;S6;A;L;M;P;M1;ER;UN;UN2;TR1;TR2;TR3
[1]   TOOLCONT
     ∇


     ∇TOOLCONT2;STAT;STOW;QTY
```

```
[1]   TOOLCONT1
     ∇

     ∇TOOLINFOGET OBJ;B;MSG;C
[1]   B←'TOOL',(⍕TOOLNN[OBJ]),'INFO'⋄→(TWO≠⎕NC B)/B1⋄C←⍎B
[2]   B3:MSG←(DLBS2 ,TOOLLIST[OBJ+ONE;]),' Usage Instructions'
[3]   C←((TEN,⌈SIX+1.25×⍴MSG)⌈⍴C)↑C
[4]   C←Z1 C⋄C←⍉DLBS3 ⍉DLBS3 C⋄→(ZERO=×/⍴C)/B2⋄⍎B,'←C'⋄→ZERO
[5]   B1:C←0 0⍴''⋄→B3
[6]   B2:C←⎕EX B
     ∇

     ∇TOOLSEARCH OBJ;ST;G1;BINS;VAL;MSG;MSG2;MSG3;H1;H2;H3
[1]   SEARCHCTL OBJ
     ∇

     ∇TOOLSTOWGET OBJ;QTY;F;VOLS
[1]   →(TOOLVOL[OBJ]≠TOOLVOLDEF)/B1
[2]   ERRORWINDOW 'Effective Volume has not been defined for ',,TOOLLIST[OBJ+1;]
[3]   →ZERO
[4]   B1:F←-TOOLNN[OBJ]⋄QTY←TOOLUSE2[TOOLUSE1⍳F]
[5]   VOLS←(STOWVOL<TOOLVOL[OBJ])/(⍴STOWVOL)↑STOWNN⋄TOOLCONT2
[6]   →(QTY=TOOLUSE2[TOOLUSE1⍳F])/ZERO
[7]   (TOOLUSEDEF CVTNUM1 TOOLUSE2[TOOLUSE1⍳F]) CLIST1 NONE↓RECT[3;]
     ∇

     ∇TOOLUSEGET OBJ;G1;G2;G3;G4;C;D;J;E;H;VAL;R;K;F;G5;A;B;MSG2;HELPMSG
[1]   F←TOOLNN[OBJ]⋄G1←TOOLUSE1⍳-F⋄G2←NONE+((G1↓TOOLUSE1)<ZERO)⍳ONE
[2]   J←G2↑G1↓TOOLUSE1⋄E←G2↑G1↓TOOLUSE2⋄C←E,((⍴JOBLIST)[ONE]-G2+ONE)⍴TOOLUSEDEF
[3]   G3←JOBNN⍳J⋄H←JOBLIST[ONE+G3;]⍪JOBLIST ELIM ONE,ONE+G3⋄D←⍴C
[4]   G4←⍕(D,ONE)⍴C⋄G4[(C=TOOLUSEDEF)/⍳D;]←' '⋄H←H,'/',G4,' '
[5]   HELPMSG←XTOOLUSEHELP
[6]   VAL←'ENTER NUMBER REQUIRED FOR THIS TASK:  '⋄H[(E=NONE)/⍳G2;ONE↓⍴H]←'*'
[7]   MSG2←''⋄→(ZERO>TOOLUSE2[G1])/B1⋄MSG2←'Quantity In Stock = ',⍕TOOLUSE2[G1]
[8]   B1:R←C EDA 'H VAL Tool Usage For ',DLBS2 TOOLLIST[IZ⍴ONE+OBJ;]
[9]   TOOLUSE2[G1+⍳G2]←G2↑R⋄R←G2↓R⋄A←(R≠TOOLUSEDEF)/R
[10]  G5←(R≠TOOLUSEDEF)/(⍳NONE+(⍴JOBLIST)[ONE]) ELIM G3⋄G2←G1+G2
[11]  TOOLUSE1←(G2↑TOOLUSE1),JOBNN[G5],G2↓TOOLUSE1
[12]  TOOLUSE2←(G2↑TOOLUSE2),A,G2↓TOOLUSE2⋄G1←(TOOLUSE2>ZERO)∨TOOLUSE1<ZERO
[13]  TOOLUSE1←G1/TOOLUSE1⋄TOOLUSE2←G1/TOOLUSE2
     ∇

     ∇TOOLVOLGET OBJ;A;B
[1]   A←(TOOLSTOW⍳-TOOLNN[OBJ])↓TOOLSTOW⋄A←(NONE+(A<ZERO)⍳ONE)↑A
[2]   B←⌊/STOWVOL[STOWNN⍳A]
[3]   B4:A←(4 1 0 0,B) GETINPUT NONE↓,RECT[L;]⋄→(OPT=2 3 6)/B1,B2,B3
[4]   TOOLVOL[OBJ]←A⋄→ZERO
[5]   B1:(TOOLVOLDEF CVTNUM TOOLVOL[OBJ]) CLIST NONE↓,RECT[L;]⋄→ZERO
[6]   B2:TOOLVOL[OBJ]←TOOLVOLDEF⋄→B1
[7]   B3:HELPWINDOW XTOOLVOLHELP⋄→B4
     ∇
```

```
    ∇R←TOOLW;SROW;K;G1;G2;H;SCOL;OPT;N;VAL;J
[1]   R←ZERO◇SROW←⌈(M[ONE]-ONE+S1)÷ELEVEN◇→(SROW<ONE)/ZERO◇→(TWO≥SROW+TC)/ZERO
[2]   →((SROW+TC)>(ρWT)[ONE])/ZERO◇K←ONE⌈⌈(M[TWO]-S2+TEN)÷SIX
[3]   H←+\AV125=WT[ONE;]◇SCOL←ONE+H[K+LC]◇G1←SIX×ZERO⌈((SCOL≠ONE)×HιSCOL-ONE)-LC
[4]   G2←(S4-S2+NINE)⌊ONE+SIX×(HιSCOL)-LC+ONE
[5]   INVERTRECT D+(ELEVEN×SROW),G1,(ELEVEN×SROW),G2◇SROW←SROW+TC
[6]   →(SROW=(ρWT)[ONE])/B3◇→(L1,L2,L3)[SCOL]
[7]   L1:N←DLBS2 ,TOOLLIST[OBJ+ONE;]
[8]   H←~YESNOWINDOW 'Do You Wish To Delete This Copy Of ',N,'?'◇→H/B1
[9]   TOOLUSE2[TOOLUSE1ιF]←QTY-ONE◇TOOLSTOW←TOOLSTOW ELIM NTWO+SROW+TOOLSTOWιF
[10]  TOOLSTAT←TOOLSTAT ELIM NTWO+SROW+TOOLSTATιF◇TR1←TR1 ELIM SROW
[11]  TOOLUSERS←TOOLUSERS ELIM NTWO+SROW+TOOLUSERSιF
[12]  TOOLUSEMODS←TOOLUSEMODS ELIM NTWO+SROW+TOOLUSEMODSιF
[13]  TOOLUSEX←TOOLUSEX ELIM NTWO+SROW+TOOLUSEXιF
[14]  TR2←TR2 ELIM SROW◇TR3←TR3 ELIM SROW
[15]  STOW←STOW ELIM SROW-TWO◇STAT←STAT ELIM SROW-TWO◇QTY←QTY-ONE◇→B2
[16]  L2:H←(6 5 1 0,VOLS) REPLYWINDOW 'Enter New Compartment Name:'◇→(0=ρH)/B1
[17]  TOOLSTOW[NTWO+SROW+TOOLSTOWιF]←STOWNN[H]◇STOW[NTWO+SROW]←STOWNN[H]
[18]  TR2←(H2TOOL CAPPEND STOWLIST[ONE+STOWNNιSTOW;]),AV125◇→B2
[19]  L3:VAL←'STAT'◇H←7 1 REPLYWINDOW 'Enter Tool Status:'◇→(∨/OPT=2 3 6)/B1
[20]  TOOLSTAT[NTWO+SROW+TOOLSTATιF]←STATNN[H]◇STAT[SROW-TWO]←STATNN[H]
[21]  B4:TR3←H3TOOL CAPPEND STATLIST[STATNNιSTAT;]
[22]  B2:WT←TR1,TR2,TR3◇WT[TWO;]←'='◇WT←WT APPEND 'CREATE NEW COPY'
[23]  ERASERECT ER◇R←ONE◇→ZERO
[24]  B5:ERRORWINDOW 'You Cannot Delete The Last Copy Of A Tool'
[25]  B1:INVERTRECT D+(ELEVEN×SROW-TC),G1,(ELEVEN×SROW-TC),G2◇→ZERO
[26]  B3:H←(6 5 1 0,VOLS) REPLYWINDOW 'Enter Compartment Name:'◇→(ZERO=ρH)/B1
[27]  VAL←'STAT'◇J←7 1 REPLYWINDOW 'Enter Tool Status:'◇→(∨/OPT=2 3 6)/B1
[28]  K←(TOOLSTOWιF)+ρSTOW◇TOOLSTOW←(K↑TOOLSTOW),STOWNN[H],K↓TOOLSTOW
[29]  K←(TOOLSTATιF)+ρSTAT◇TOOLSTAT←(K↑TOOLSTAT),STATNN[J],K↓TOOLSTAT
[30]  TOOLUSERS←(K↑TOOLUSERS),ZERO,K↓TOOLUSERS
[31]  TOOLUSEX←(K↑TOOLUSEX),ZERO,K↓TOOLUSEX
[32]  TOOLUSEMODS←(K↑TOOLUSEMODS),FIVE,K↓TOOLUSEMODS
[33]  QTY←QTY+ONE◇TOOLUSE2[TOOLUSE1ιF]←QTY◇STOW←STOW,STOWNN[H]
[34]  STAT←STAT,STATNN[J]
[35]  TR1←(H1TOOL CAPPEND ((QTY,FIVE)ρ'COPY '),⍕(QTY,ONE)ριQTY),AV125
[36]  TR2←(H2TOOL CAPPEND STOWLIST[ONE+STOWNNιSTOW;]),AV125◇→B4
    ∇
```

```
    ∇R←UPPERCASE A;B;C
[1]   C←,A◇B←(C∊ALFL)/ιρC◇C[B]←ALFU[ALFLιC[B]]◇R←(ρA)ρC
    ∇
```

```
    ∇W←WS
[1]   W←⎕WSID◇W←(-(⌽W)ι':')↓W◇W←(ONE-(⌽W)ι':')↓W
    ∇
```

```
    ∇XCREWFORMFILL A
[1]   (NONE CVINUM1 CREWNN[A]) CLIST1 NONE↓RECT[TWO;]
[2]   (CREWBDDEF CVIDATE1 CREWBD[A]) CLIST1 NONE↓RECT[THREE;]
[3]   (CREWMSDEF CVINUM1 CREWMS[A]) CLIST1 NONE↓RECT[FOUR;]
[4]   (CREWHTDEF CVINUM1 CREWHT[A]) CLIST1 NONE↓RECT[FIVE;]
[5]   (CREWASDEF[ONE] CVITIME1 CREWAS[ONE;A]) CLIST1 NONE↓RECT[SIX;]
[6]   (CREWASDEF[TWO] CVITIME1 CREWAS[TWO;A]) CLIST1 NONE↓RECT[SEVEN;]
[7]   CREWRANKLIST[CREWRANK[A];] CLIST1 NONE↓RECT[EIGHT;]
    ∇
```

```
    ∇XJOBFORMFILL A
[1]   (NONE CVTNUM1 JOBNN[A]) CLIST1 NONE↓RECT[TWO;]
[2]   (JOBTIMEDEF CVTNUM1 PERFDATA2[PERFDATA1⍳-JOBNN[A]]) CLIST1 NONE↓RECT[3;]
[3]   (JOBSIZEDEF CVTNUM1 JOBSIZE[A]) CLIST1 NONE↓RECT[FOUR;]
[4]   (JOBPOWERDEF CVTNUM1 JOBPOWER[A]) CLIST1 ¯1↓RECT[FIVE;]
[5]   (JOBMULTIPLEXDEF CVTNUM1 JOBMULTIPLEX[A]) CLIST1 NONE↓RECT[SIX;]
[6]   (JOBMEMORYDEF CVTNUM1 JOBMEMORY[A]) CLIST1 NONE↓RECT[SEVEN;]
[7]   (JOBTRANSDEF CVTNUM1 JOBTRANS[A]) CLIST1 NONE↓RECT[EIGHT;]
[8]   JOBAUDIOLIST[JOBAUDIO[A];] CLIST1 NONE↓RECT[NINE;]
    ∇


    ∇XMANFORMFILL A
[1]   (NONE CVTNUM1 MANNN[A]) CLIST1 NONE↓RECT[TWO;]
[2]   (MANASDEF[ONE] CVTTIME1 MANAS[ONE;A]) CLIST1 NONE↓RECT[THREE;]
[3]   (MANASDEF[TWO] CVTTIME1 MANAS[TWO;A]) CLIST1 NONE↓RECT[FOUR;]
[4]   (MANPOWERDEF CVTNUM1 MANPOWER[A]) CLIST1 NONE↓RECT[FIVE;]
[5]   (MANMULTIPLEXDEF CVTNUM1 MANMULTIPLEX[A]) CLIST1 NONE↓RECT[SIX;]
[6]   (MANMEMORYDEF CVTNUM1 MANMEMORY[A]) CLIST1 NONE↓RECT[SEVEN;]
[7]   (MANTRANSDEF CVTNUM1 MANTRANS[A]) CLIST1 NONE↓RECT[EIGHT;]
    ∇


    ∇XSTOWFORMFILL A;C
[1]   (NONE CVTNUM1 STOWNN[A]) CLIST1 NONE↓RECT[TWO;]
[2]   (NONE CVTNUM1 STOWVOL[A]) CLIST1 NONE↓RECT[THREE;]
[3]   →(STOWMOD[A]=STOWMODDEF)/B1
[4]   C←'MODULE ',(⍕STOWMOD[A]),': ',MODNAMES[ONE+MODMATRIX[IZ⍴STOWMOD[A];FOUR];]
[5]   C CLIST1 NONE↓RECT[FOUR;]⋄→B2
[6]   B1:(STOWMODDEF CVTNUM1 STOWMODDEF) CLIST1 NONE↓RECT[FOUR;]
[7]   B2:(STOWXYZDEF CVTNUM1 STOWX[A]) CLIST1 NONE↓RECT[FIVE;]
[8]   (STOWXYZDEF CVTNUM1 STOWTHETA[A]) CLIST1 NONE↓RECT[SIX;]
    ∇


    ∇XTOOLFORMFILL A
[1]   (NONE CVTNUM1 TOOLNN[A]) CLIST1 NONE↓RECT[TWO;]
[2]   (TOOLUSEDEF CVTNUM1 TOOLUSE2[TOOLUSE1⍳-TOOLNN[OBJ]]) CLIST1 NONE↓RECT[3;]
[3]   (TOOLVOLDEF CVTNUM1 TOOLVOL[A]) CLIST1 NONE↓RECT[FOUR;]
    ∇


    ∇R←YESNOWINDOW A;OS;K;M;D;UN;UN2;P;KK
[1]   D←22 20 70 480⋄UN←22 20 70 484⋄UN2←⎕GETBITS UN⋄ERASERECT D⋄KK←ONE
[2]   FRAMERECT 2 4⍴D,24 22 68 478⋄TEXTSIZE 18⋄TEXTFACE 65⋄MOVETO 52 419
[3]   FRAMEROUNDRECT 2 6⍴33 395 57 470 15 15 31 393 59 472 17 17⋄DRAWTEXT 'NO'
[4]   FRAMEROUNDRECT 2 6⍴33 310 57 385 15 15 31 308 59 387 17 17
[5]   MOVETO 52 325⋄DRAWTEXT 'YES'⋄⎕SOUND BEEP⋄TEXTSIZE NINE⋄TEXTFACE ONE
[6]   K←35+FIVE×FOUR-THREE⌊⌈(⍴A)÷39
[7]   B1:MOVETO K,35⋄OS←39-(⌽39↑A)⍳' '⋄P←OS<ZERO⋄OS←OS+39×P⋄→((KK=3)∧OS<⍴A)/B8
[8]   DRAWTEXT OS↑A⋄KK←KK+ONE⋄A←(OS+~P)↓A⋄K←K+ELEVEN⋄→(ZERO≠⍴A)/B1⋄TEXTFACE ZERO
[9]   B2:SHOWCURSOR⋄M←⎕GETKEY⋄→(THREE≠⍴M)/B2⋄→(TWO=M[ONE])/B2⋄→(33>M[TWO])/B3
[10]  →(57<M[TWO])/B3⋄→(395>M[THREE])/B6⋄→(470<M[THREE])/B6⋄R←ZERO
[11]  B7:UN2 ⎕PUTBITS UN
[12]  B4:→(ZERO≠⍴⎕GETKEY)/B4⋄→ZERO
[13]  B3:SHOWCURSOR⋄→(BUTTON)/B3⋄→B2
[14]  B6:→(310>M[THREE])/B3⋄→(385<M[THREE])/B3⋄R←ONE⋄→B7
[15]  B8:DRAWTEXT (35↑A),'...'⋄TEXTFACE ZERO⋄→B2
    ∇
```

```
∇R←Z C;A;CR;S1;S2;S3;S4;G1;G2;K;G3;G4;G5;RECT;TR;M;M1;B;LC;RC;TC;BC;G5;HS;P
 TEXTSIZE 12◊ TEXTFONT ZERO◊TEXTFACE ZERO◊MSG← ' ',MSG,' '◊MS←TEXTWIDTH MSG
 L←ZERO◊LC←L◊TC←L◊WTEXT←C◊D←LC◊F←TC◊SIZE←175 375⌊11 6×1 2+⍴C◊N←1 3
 ULC←100 100◊P←⍳11◊C1←L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11
 B4:S1←ULC[ONE]◊S2←ULC[TWO]◊S3←S1+SIZE[ONE]◊S4←S2+SIZE[TWO]
 BC←⌊(NTWO+SIZE[ONE])÷11◊RC←⌊(¯9+SIZE[TWO])÷SIX◊OUT←(BC,RC)↑(TC,LC)↓WTEXT
 F←TC⌈(NONE+TC+BC)⌊F◊D←LC⌈(NONE+LC+RC)⌊D◊CURP←(S1+TWO+11×F-TC),S2+7+6×D-LC
 RECT←S1,S2,S3,S4◊B←¯18 0 18 18+RECT◊UN←TWO↑B◊UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
 K←⎕EX 'UN2'◊UN2←⎕GETBITS UN◊TR←¯18 0 1 18+S1,S2,S1,S4◊ERASERECT B
 CR←¯14 10 ¯3 22+S1,S2,S1,S2◊A←¯14 2 ¯3 14+S1,S4,S1,S4
 G1←0 ¯1 18 18+S1,S4,S1,S4◊G1 DRAWPICTURE UPARROW
 G2←¯18 ¯1 0 18+S3,S4,S3,S4◊G2 DRAWPICTURE DOWNARROW
 G3←¯1 0 18 18+S3,S2,S3,S2◊G3 DRAWPICTURE LEFTARROW
 G4←¯1 ¯18 18 0+S3,S4,S3,S4◊G4 DRAWPICTURE RIGHTARROW
 G5←¯1 ¯1 18 18+S3,S4,S3,S4◊G5 DRAWPICTURE CORNER
 K←(S1-15)+TWO×⍳SIX◊K←⍉4 6⍴(K-ONE),(SIX⍴TWO+S2),K,SIX⍴S4+16
 RECT←RECT⍪10 4⍴CR,A,TR,G1,G2,G3,G4,G5◊FRAMERECT K⍪B⍪¯2 0↓RECT
 G1←⌊PFIVE×15+S4+S2-MS⌊S4-S2+30◊K←(S1-17),G1,(S1-ONE),18+S4+S2-G1
 K←K⍪3 4⍴(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
 ERASERECT K⍪2 4⍴0 ¯1 1 0 0 0 1 1+A[1 2 3 2 1 4 3 4]◊MOVETO (S1-THREE),G1
 TEXTSIZE 12◊TEXTFONT ZERO◊MSG DRAWMSG MS,MS⌊S4-S2+30◊TEXTNORMAL
 A←3 4⍴(A+1 1 ¯1 ¯1),0 18 17 ¯18 18 0 ¯18 17+S3,S2,S3,S4,S1,S4,S3,S4
 GRAYPAT FILLRECT A
 B14:MOVETO 10 8+S1,S2◊DRAWTEXT OUT◊→(L=11)/B18
 B17:G1←(BC×¯36+S3-S1)÷2×1↑⍴WTEXT◊G2←S1+18+((TC+.5×BC)÷1↑⍴WTEXT)×(S3-S1)-36
 G1←(⌊G2-G1),S4,((S3-18)⌊⌊G2+G1),17+S4◊ERASERECT G1◊FRAMERECT G1
 RECT[TEN;]←G1◊→(L=5 6 10)/B2,B2,B2
 B18:G5←(RC×¯36+S4-S2)÷TWO×1↓⍴WTEXT◊G2←S2+18+((LC+.5×RC)÷1↓⍴WTEXT)×S4-S2+36
 G5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◊ERASERECT G5◊FRAMERECT G5
 RECT[11;]←G5
 B2:MOVETO CURP◊PENMODE TEN◊K←,WTEXT[ONE+F;]
 G←(S1+2+11×F-TC),(S2+TWO+SIX×D-LC),(S1+12+11×F-TC),(S2+EIGHT+SIX×RC),6 0
 B44:LINETO CURP+7 0◊E←⎕GETKEY◊→(N=⍴E)/B40,B41◊E←⎕GETKEY◊→(N=⍴E)/B40,B41
 E←⎕GETKEY◊→(N=⍴E)/B40,B41◊E←⎕GETKEY◊→(N=⍴E)/B40,B41◊E←⎕GETKEY
 →(N=⍴E)/B40,B41◊E←⎕GETKEY◊→(N=⍴E)/B40,B41◊LINETO CURP◊E←⎕GETKEY
 →(N=⍴E)/B42,B43◊E←⎕GETKEY◊→(N=⍴E)/B42,B43◊E←⎕GETKEY◊→(N=⍴E)/B42,B43
 E←⎕GETKEY◊→(N=⍴E)/B42,B43◊E←⎕GETKEY◊→(N=⍴E)/B42,B43
 E←⎕GETKEY◊→(N=⍴E)/B42,B43◊→B44
 B40:LINETO CURP
 B42:→(13 8=E)/B45,B46◊→(D=LC+RC)/B49◊→(E>255)/B44◊KK←AV[E+ONE]
 K←(D↑K),KK,D↓K◊D←D+ONE◊G[TWO]←G[TWO]+SIX◊CURP←CURP+0 6◊MOVETO 8 0+G[1 2]
 SCROLLRECT G◊PENMODE EIGHT◊DRAWTEXT KK◊PENMODE TEN◊MOVETO CURP◊→B44
 B45:→(F=BC+TC-1)/B49◊L←(⍴WTEXT)[TWO]◊→(L<⍴K)/B55◊WTEXT[IZ⍴ONE+F;]←L↑K◊→B56
 B55:WTEXT←((F,⍴K)↑WTEXT)⍪K⍪((ONE+F-(⍴WTEXT)[ONE]),⍴K)↑WTEXT
 B56:CURP←(11+ONE↑CURP),S2+SEVEN◊D←LC◊F←F+ONE◊→((F+ONE)≤ONE↑⍴WTEXT)/B2
 WTEXT←WTEXT⍪' '◊→B2
 B46:→(D=LC)/B49◊D←D-ONE◊K←(D↑K),(D+ONE)↓K◊CURP←CURP-0 6◊PENMODE EIGHT
 G[FIVE]←¯6◊SCROLLRECT G◊G[TWO]←G[TWO]-SIX◊G[FIVE]←SIX
 MOVETO (S1+NONE+ELEVEN×ONE+F-TC),S2+TWO+SIX×RC◊DRAWTEXT NONE↑(RC+LC)↑K
 PENMODE TEN◊MOVETO CURP◊→B44
 B49:⎕SOUND BEEP◊MOVETO CURP◊→B44
 B41:LINETO CURP
 B43:L←(⍴WTEXT)[TWO]◊→(L<⍴K)/B53◊WTEXT[IZ⍴1+F;]←L↑K◊→B54
 B53:WTEXT←((F,⍴K)↑WTEXT)⍪K⍪((F+ONE-(⍴WTEXT)[ONE]),⍴K)↑WTEXT
 B54:PENMODE EIGHT◊→(TWO=E[ONE])/B2◊M←E[2 3]
 L←ONE↑(M PTINRECT RECT)/P◊→(ZERO=L)/B2◊→C1[L]
 B3:⎕SOUND BEEP◊→B2
 L1:CURP←(NONE+BC,RC)⌊0 0⌈⌊(M-(S1+TWO),S2+SEVEN)÷11 6◊D←LC+CURP[TWO]
 F←TC+CURP[ONE]◊CURP←(2 7+S1,S2)+11 6×CURP◊→B2
 L2:→BUTTON/L2◊→(~GETMOUSE PTINRECT RECT[L;])/B2◊UN2 ⎕PUTBITS UN◊R←WTEXT◊→0
 L3:→BUTTON/L2◊→(~GETMOUSE PTINRECT RECT[3;])/B2◊UN2 ⎕PUTBITS UN◊R←C◊→0
```

```
[61]  L4:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
[62]  B7:→(~BUTTON)/B5◊FRAMERECT M1◊M1←B+4ρ(0 0⌈GETMOUSE)-M◊FRAMERECT M1◊→B7
[63]  B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
[64]  B30:FRAMERECT M1◊UN2 ⎕PUTBITS UN◊PENPAT BLACKPAT◊PENMODE EIGHT◊→B4
[65]  L5:→(TC≤ZERO)/B3◊SCROLLRECT ((1 7,(ONE+11×BC),NTWO)+RECT[1;1 2 1 4]),0 11
[66]  MOVETO 10 8+S1,S2◊DRAWTEXT RC↑LC↓,WTEXT[TC;]◊TC←TC-ONE◊F←F⌊NONE+TC+BC
[67]  B50:GRAYPAT FILLRECT G1◊CURP[ONE]←S1+TWO+ELEVEN×F-TC◊→B17
[68]  L6:→(TC≥(ρWTEXT)[ONE]-BC)/B3◊TC←TC+ONE
[69]  SCROLLRECT ((1 7,(ONE+ELEVEN×BC),NTWO)+RECT[ONE;1 2 1 4]),0 ¯11
[70]  MOVETO (S1+NONE+ELEVEN×(ρOUT)[ONE]),S2+EIGHT◊DRAWTEXT RC↑LC↓,WTEXT[BC+TC;]
[71]  F←F⌈TC◊→B50
[72]  L7:→(LC≤ZERO)/B3◊G3←LC⌊HSCROLL
[73]  SCROLLRECT ((1 7 ¯1,SEVEN+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO
[74]  LC←LC-G3◊OUT←(BC,G3⌊RC)↑(TC,LC)↓WTEXT
[75]  MOVETO 10 8+S1,S2◊DRAWTEXT OUT◊D←D⌊NONE+RC+LC
[76]  B51:GRAYPAT FILLRECT G5◊CURP[TWO]←S2+SEVEN+SIX×D-LC◊→B18
[77]  L8:G3←(ρWTEXT)[TWO]-RC◊→(LC≥G3)/B3◊G3←HSCROLL⌊G3
[78]  SCROLLRECT ((1 7 ¯1,SEVEN+SIX×RC)+RECT[ONE;1 2 3 2]),(¯6×G3),ZERO
[79]  LC←LC+G3◊OUT←(BC,-G3)↑(BC,RC)↑(TC,LC)↓WTEXT
[80]  MOVETO (S1+10),S2+EIGHT+SIX×RC-G3◊DRAWTEXT OUT◊D←D⌈LC◊→B51
[81]  L9:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
[82]  B8:→(~BUTTON)/B9◊FRAMERECT M1
[83]  M1←(2↑B),(18 18+ULC)+36 36⌈SIZE+GETMOUSE-M◊FRAMERECT M1◊→B8
[84]  B9:SIZE←36 36⌈SIZE+GETMOUSE-M◊→B30
[85]  L10:M1←G1◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[86]  B21:→(~BUTTON)/B22◊FRAMERECT M1
[87]  K←(18+S1-G1[1])⌈(S3-18+G1[3])⌊1↑GETMOUSE-M◊M1←G1+4ρK,0◊FRAMERECT M1◊→B21
[88]  B22:FRAMERECT M1◊GRAYPAT FILLRECT G1◊TC←TC+⌊PFIVE+K×(ρWTEXT)[ONE]÷S3-S1+36
[89]  OUT←(BC,RC)↑(TC,LC)↓WTEXT◊ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]
[90]  F←TC⌈(NONE+TC+BC)⌊F◊CURP[ONE]←S1+TWO+ELEVEN×F-TC
[91]  B31:PENPAT BLACKPAT◊PENMODE EIGHT◊→B14
[92]  L11:M1←G5◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[93]  B23:→(~BUTTON)/B24◊FRAMERECT M1
[94]  K←(18+S2-G5[2])⌈(S4-18+G5[4])⌊1↓GETMOUSE-M◊M1←G5+4ρ0,K◊FRAMERECT M1◊→B23
[95]  B24:FRAMERECT M1◊GRAYPAT FILLRECT G5
[96]  LC←LC+⌊PFIVE+K×(ρWTEXT)[TWO]÷S4-S2+36◊OUT←(BC,RC)↑(TC,LC)↓WTEXT
[97]  ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◊D←LC⌈(¯1+LC+RC)⌊D◊CURP[2]←S2+7+6×D-LC◊→B31
    ∇


    ∇R←Z1 H;A1;A2;D;CURP;E;F;L;MS;OUT;WTEXT;SIZE;ULC;UN;UN2;C1;N;G;KK
[1]   R←Z H
    ∇


      )VARS
ACTIONSTOP        ALFL              ALFU              APLICON
APLICONEXE        AV                AV125             BEEP
BINH1             BINH2             BINH3             BINH4
BINH5             BINH6             BINH7             BLACKPAT
COMLIST           CORNER            CREW17INFO        CREW3INFO
CREW5INFO         CREW9INFO         CREWAS            CREWASDEF
CREWBD            CREWBDDEF         CREWHT            CREWHTDEF
CREWLIST          CREWMS            CREWMSDEF         CREWN
CREWNN            CREWRANK          CREWRANKDEF       CREWRANKLIST
CREWRANKN         CREWRANKNN        DESKTOP           DOWNARROW
EIGHT             ELEVEN            EXITMSG           FIVE
FOUR              GRAYPAT           H1STOW            H1TOOL
H2TOOL            H3TOOL            HSCROLL           IZ
JOB3INFO          JOBAUDIO          JOBAUDIODEF       JOBAUDIOLIST
JOBAUDION         JOBAUDIONN        JOBLIST           JOBMEMORY
JOBMEMORYDEF      JOBMULTIPLEX      JOBMULTIPLEXDEF   JOBN
```

JOBNN	JOBPOWER	JOBPOWERDEF	JOBSIZE
JOBSIZEDEF	JOBTIMEDEF	JOBTRANS	JOBTRANSDEF
LEFTARROW	LOADRECT	LOGO	MANAS
MANASDEF	MANCREW	MANJOB	MANLIST
MANMEMORY	MANMEMORYDEF	MANMULTIPLEX	MANMULTIPLEXDEF
MANN	MANNN	MANPOWER	MANPOWERDEF
MANTRANS	MANTRANSDEF	MFIVEHEADER	MFIVEICON2
MODLENGTH	MODMATRIX	MODNAMES	MONTHS
MOS	MOS2	NFIVE	NFOUR
NHA	NINE	NONE	NTWO
NULL	ONE	OPTIONS	OPTIONSN
PERFDATA1	PERFDATA2	PFIVE	PRIMEDISK
REVISENAMEHELP	RIGHTARROW	SCHEDDISK	SCHEDWS
SCREEN	SEARCHCMATRIX	SEVEN	SIX
STATLIST	STATN	STATNN	STOWLIST
STOWMOD	STOWMODDEF	STOWN	STOWNN
STOWTHETA	STOWVOL	STOWVOLDEF	STOWX
STOWXYZDEF	TEN	THREE	TOOL11INFO
TOOLHIST	TOOLLIST	TOOLN	TOOLNN
TOOLSTAT	TOOLSTOW	TOOLUSE1	TOOLUSE2
TOOLUSEDEF	TOOLUSEMODS	TOOLUSERS	TOOLUSEX
TOOLVOL	TOOLVOLDEF	TWO	UPARROW
WHITEPAT	XCREWASHELP	XCREWBDHELP	XCREWFORM
XCREWFORMRECTS	XCREWFORMSIZE	XCREWHTHELP	XCREWICON
XCREWICONEXE	XCREWMSHELP	XCREWOPTS	XCREWPERFHELP
XJOBDEFHELP	XJOBFORM	XJOBFORMRECTS	XJOBFORMSIZE
XJOBICON	XJOBICONEXE	XJOBMEMORYHELP	XJOBMULTIPLEXHELP
XJOBOPTS	XJOBPERFHELP	XJOBPOWERHELP	XJOBSIZEHELP
XJOBTOOLHELP	XJOBTRANSHELP	XMANASHELP	XMANFORM
XMANFORMRECTS	XMANFORMSIZE	XMANICON	XMANICONEXE
XMANMEMORYHELP	XMANMULTIPLEXHELP	XMANOPTS	XMANPOWERHELP
XMANTRANSHELP	XSTOWFORM	XSTOWFORMRECTS	XSTOWFORMSIZE
XSTOWICON	XSTOWICONEXE	XSTOWOPTS	XSTOWTHETAHELP
XSTOWVOLHELP	XSTOWXHELP	XTOOLFORM	XTOOLFORMRECTS
XTOOLFORMSIZE	XTOOLICON	XTOOLICONEXE	XTOOLOPTS
XTOOLUSEHELP	XTOOLVOLHELP	ZERO	

C.2 The SCHED Workspace

When the SCHED workspace is loaded, the latent expression 'START' is activated, starting execution of the function START. START is a niladic function (it has no arguments) and returns no explicit result. If the SCHED workspace was entered from the database management system in the PRIME workspace, then START will load into MFIVE the flight plan data being transferred and commence scheduler operation by executing the function S. If instead the SCHED workspace was entered directly (e.g., by clicking on the SCHED icon in the Macintosh finder), then START will direct the clearing of the screen and the writing of the word MFIVE in the bar at the top of the screen. The scheduler will then execute the function SL which will prompt the user for the name of a previously stored scheduling problem which is to be loaded in.

C.2.1 The Function INIT

The function INIT oversees operation of the scheduler. The function INIT is called from the function INIT4, which is called from the function INIT3, which is called from the function INIT2, which is called from the function INIT1, with is called by both S and SL. INIT1, INIT2, INIT3, and INIT4 are "shielding functions": their only purpose is to allow the declaration of more local variables than could comfortably be included in the header of INIT. INIT has a single argument R, which is a character vector pointer to the names of the variables which contain the problem to be scheduled. If R is a null vector, then the user is prompted for the name of a schedule to be loaded into memory.

The function INIT draws the scheduling information on the display, takes action when the mouse is clicked on sensitive areas on the screen, and monitors and acts on menu selections. The mouse is monitored at the line labelled B11. When the mouse is clicked or a menu item is selected, function control passes to the following branch points:

Branch Point	Action Which Causes Branching
L1	Clicking on up arrow (to increase ruler scale by 1 minute/pixel)
L2	Clicking on down arrow (to decrease ruler scale by 1 minute/pixel)
L3	Clicking on left arrow (to decrease ruler origin)
L4	Clicking on right arrow (to increase ruler origin)
L5	Clicking on box with two rectanges (to manually change ruler scale and origin)

L6	Clicking on one of the resource boxes (to enlarge and inspect resource usage display)
L10	Clicking and moving the rectangle corresponding to a crewmembers performance of a job being scheduled (to manually assign the crewmember to a job)
L11	Clicking on the Job Information window
L12	Clicking on a crewmember's crew number (e.g., **C1**) (to assign a job to a crewmember at the crewmembers earliest available start time)
L13	Clicking on **C0**
B1	Selecting REDRAW SCREEN from the OPTIONS menu
B14	Clicking inside one of the crewmembers scheduling bars (to identify a job or constraint)
B47	Selecting UNSCHEDULE ALL or RESTART SCHEDULE from the AUTOPLAN menu
B53	Selecting COMPLETE SCHEDULE from the AUTOPLAN menu
A1	Selecting DISPLAY JOB INFO from the OPTIONS menu (to open or close the Job Information window)
A2	Selecting AUTO JOB SELECT from the AUTOPLAN menu
A3	Selection SELECT JOB from the AUTOPLAN menu
A4	Selecting AUTO CREW SELECT from the AUTOPLAN menu
A5	Selecting SELECT CREW from the AUTOPLAN menu
A6	Selecting SAVE SCHEDULE from the OPTIONS menu
A7	Selecting LOAD SCHEDULE from the OPTIONS menu
A8	Selecting DELETE SCHEDULE from the OPTIONS menu
A9	Selecting PRINT SCHEDULE from the OPTIONS menu
A10	Selecting RETURN TO APL from the OPTIONS menu
A11	Selecting TIME CONSTRAINTS from the CONSTRAINTS menu
A12	Selecting TARGETS from the CONSTRAINTS menu
A13	Selecting anything from the MOUSE MODE menu (to change the MOUSE MODE)
A15	Selecting SET HEURISTIC from the AUTOPLAN menu
A16	Selecting SET SEARCH PARAMETERS from the AUTOPLAN menu
A20	Selecting PRIORITY LEVELS from the CONSTRAINTS menu

C.2.2 Global Variables in the SCHED Workspace

The variables ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, and ELEVEN all contain the scalar integer indicated by their name. The variables NONE, NTWO, NTHREE, NFOUR, NFIVE, NSIX, NSEVEN, NEIGHT, and NNINE contain the integers -1, -2, -3, -4, -5, -6, -7, -8, and -9 respectively. The variable PFIVE contains the number .5. The variable IZ contains a vector of zero length.

The variable AV contains the APL character set. ALFL contains the lower case alphabet and ALFU contains the upper case alphabet. AV125 contains the 125 character in AV, which is the "|" character. This character is used as a cursor and to separate fields in windows produced by MFIVE.

The variable ACTIONSTOP contains instructions on what should happen when program interruption occurs. When ACTIONSTOP is null, control will return to the APL environment.

The variable BEEP specifies the tone heard whenever MFIVE produces a beep sound.

The variables BLACKPAT, GREYPAT, WHITEPAT, BARPAT, and STRIPEPAT contain the pen patterns for producing black, grey, white, barred and striped drawings on the screen. The variable PAT is a matrix containing the patterns for white, grey and black.

The variables CORNER, RIGHTARROW, LEFTARROW, UPARROW, and DOWNARROW contain the pictures for components of the windows which MFIVE draws.

The variable MONTHS is a twelve row character matrix with each row containing the first three letters of the name of one of the twelve months of the year. The variable MOS is a 24 x 1 column matrix. The first twelve rows contain the cumulative number of days through the end of each month in a non-leap year. The second twelve rows contain the cumulative number of days through the end of each month in a leap year.

The variable DESKTOP contains the message that MFIVE does not support desk accessories. The variable NHA contains the message "No help is currently available."

The variable MFIVEHEADER contains the information necessary to draw the word MFIVE in the bar at the top of the screen.

The variable LOADRECT contains the size of the window presented when loading, saving, or deleting a database. The variable SCREEN contains the size of the Macintosh screen that is used by MFIVE.

The variable OPTIONS contains the text for the OPTIONS menu. The variable OPTIONSN contains the menu number for the OPTIONS menu. The variable PLANMENU contains the text for the AUTOPLAN menu. The variable PLANN contains the menu number for the AUTOPLAN menu. The variable CONSTMENU contains the text for the CONSTRAINTS menu. The variable CONSTN contains the menu number for the constraints menu. The variable MOUSEMENU contains the text for the MOUSE MODE menu. The variable MOUSEN contains the menu number for the MOUSE MODE menu. The variable SEARCHMENU contains the text for the SEARCH menu. The variable SEARCHN contains the menu number for the SEARCH menu.

The variables CONHELP1, CONHELP2, CONHELP3, CONHELP4, and CONHELP5 all contain help messages for the time constraint entering functions SETCON and IRC. The variable JOBATTHELP contains a help message for the numerical replanning function JOBATT. The variable PRIORITYHELP contains a help message for the function PRIORITYGET, which modifies job priorities. The variables SSHELP1, SSHELP2, and PIXHELP contain help messages for the function SETPIX, which changes the timeline scale and origin. The variables SEARCHHELP1, SEARCHHELP2, SEARCHHELP3, and SEARCHHELP4 contain help messages for the function SETSEARCH, which sets the search parameters. The variables TARHELP1 and TARHELP2 contain help messages for the function SETTAR, which sets the target constraints.

The variable JOBATTV contains the screen picture for numerical replanning (function JOBATT), and the variable ATTRECTS contains the areas within this picture which are sensitive to being clicked on. The variable AUTOPICT contains the picture displayed after selecting COMPLETE SCHEDULE from the AUTOPLAN menu (function AUTOPLAN2). The variable PRTRECTS contains the areas which are sensitive to clicking on the picture produced by selecting PRINT SCHEDULE from the OPTIONS menu (function PRTSCHED). The variable SPRECTS contains the areas of the screen which are sensitive to clicking when using the function SETPIX, which changes the timeline scale and origin. The variable SRP contains the coordinates of areas (scheduling bars) in which the crewmembers schedules are displayed, and also the coordinates of the areas above these scheduling bars where the job numbers are written. The variable SRP2 contains the coordinates of the sensitive areas on the screen corresponding to the crewmember's number. These areas are clicked on in order to assign a job to a crewmember at the crewmembers earliest possible start time).

The variable SRRECTS contains the areas of the screen which are sensitive to clicking when setting the search parameters with the function SETSEARCH. The variable TARRECTS contains the areas which are sensitive to clicking when entering target constraints (function SETTAR).

The variables C11, C12, C13, C14, C15, C16, C17, C22, C23, C24, C25, C26, C27, C33, C34, C35, C36, C37, C44, C45, C46, C47, C55, C56, C57, C66, C67, and C77 contain sets of permutations. In general, the variable Cxy contains all possible selections (one to a row, and without regard to order) of the numbers from 1 to y taken x at a time.

The variables HE0, HE1, HE2, HE3, HE4, HE5, HE6, HE7, and HE8 contain text necessary to form the time constraint window (function SETCON). The variables HEAD0, HEAD1, HEAD2, HEAD3, HEAD4, HEAD5, HEAD6, HEAD7, HEAD8, and HEAD10 all contain text necessary to form the job information window (function SETJOB).

The variables JOBMEMORYDEF, JOBMULTIPLEXDEF, JOBPOWERDEF, JOBSIZEDEF, JOBTIMEDEF, JOBTRANSDEF, and MANASDEF contain the default values (from the PRIME workspace, see Section C.1.3) for job memory usage, job multiplex usage, job power usage, job size (crew required), default job time, job transmission usage, and manifest start and end times, respectively.

The variable CONCODE contains the messages displayed to explain constraint violations when the user clicks on a blacked out region of a schedule. The variable AUDION contains the allowable audio levels (i.e., sensitive, neutral, or generating) for display in the Job Information window.

The variable PRIMEDISK contains the name of the folder (or disk, if it is not in a folder), in which the PRIME workspace is supposed to be located. The variable SCHEDDISK contains the name of the folder (or disk, if it is not in a folder), in which the SCHED workspace is supposed to be located. The variable PRIMEWS contains the name of the PRIME workspace.

The variable SCHEDDATA contains the names of all the variables which are to be saved when saving a schedule.

Variables that begin with the prefix "MAN" followed by an integer number contain flight plan information as it was transferred from the PRIME workspace. Setting the variable MAN to the prefix of a flight plan (e.g., setting MAN to be "MAN1") and then executing the function S in the APL environment will cause that flight plan to be directly loaded into the MFIVE scheduler.

C.2.3 Major Variables Used Throughout the Scheduler

In addition to the global variables described in Section C.2.2, there are many other variables which are shared by all functions called by INIT. Many of these variables describe the current state of the schedule.

Variable(s)	Description
CL	Contains the number of crewmembers in the flight plan.
ICL	Contains a vector of numbers from 1 to CL.
CRN	A matrix (CL rows) containing the names of the crewmembers.
JL	Contains the number of jobs in the flight plan.
IJL	Contains a vector of numbers from 1 to JL.
JON	A matrix (JL rows) containing the names of the jobs.
JS	Contains the number of the job being schedules, if there is one. Otherwise it is an empty vector.
JSS	This vector contains a list of jobs which have been successfully scheduled.
WL	Contains the total workload of each of the crewmembers on the jobs which have been scheduled.
JT	A vector (of length JL) containing the number of crewmembers required to perform each job.
CS	The number of crewmembers required to perform the job currently being scheduled. If no job is selected for scheduling, then this is an empty vector.
NA	A character vector containing the name of the flight plan.

MS	The start time (in minutes, from January 1, 1900 0:00) of the flight plan.
ME	The end time (in minutes, from January 1, 1900 0:00) of the flight plan.
MD	The duration of the flight plan, ME - MS, in minutes.
PD	Contains the performance times of each of the crewmembers on each of the jobs. Has JL rows and CL columns.
PDS	If a job is selected for scheduling, then this variable contains the performance times of each of the crewmembers on that job.
PR	Contains a 0-1 matrix of which crewmembers are required to perform each job because of fixed assignments. Has JL rows and CL columns. If Crewmember 1 must perform Job 7, then the entry in row 7, column 1 is a 1.
P1	Contains a 0-1 matrix of which crewmembers are assigned to each job which is currently scheduled. Has JL rows and CL columns. If Crewmember 1 is currently assigned to perform Job 7, then the entry in row 7, column 1 is a 1.
STR	Vector of length JL which contains the required start times of those jobs which have fixed start times. Otherwise, contains a -1.
ST	Vector of length JL which contains the start times of those jobs which are currently scheduled. Otherwise, contains a -1.
ST1	Vector of length JL which contains the end times of those jobs which are currently scheduled. Otherwise, contains a -1.
CT	Contains the maximum allowable differences between the start time of each job and every other job, as inferred from all time constraints entered. Also has the maximum allowable difference between the start of the flight plan and the start of each job and between the start of each job and the start of the flight plan. Has JL+1 rows and JL+1 columns. This is the ΔSMAX matrix of Section 4.2.3
CONP	Same as CT, but incorporates information about jobs which have been scheduled (by active constraint propagation, as described in Section 4.2.5).
T2	Contains the current latest start times of each of the jobs
T3	Contains the current latest end time (incorporating active constraint propagation) of each of the jobs.
TT	Contains the latest end times of each of the jobs, as inferred from all time constraints entered, but not taking into account information from jobs which have been scheduled.
MPF	The minimum performance times of each of the jobs.
MPD	Same as MPF, but updated with the actual performance time of each job as it

	becomes scheduled.
MPG	The maximum performance times of each of the jobs.
MPE	Same as MPG, but updated with the actual performance time of each job as it becomes scheduled.
PRI	The priority levels of each of the jobs.
PRC	Contains the precedence constraints.
TAR	Contains the target constraints.
AUTV	Contains, in coded form, all time constraints entered.
SP	A matrix containing the current schedule. The matrix is in the same form as Table A.30.
TL	When a job is selected for scheduling, this variable maintains information on whether or not the job is infeasible due to time, resource, or target constraints, at each time specified in SP1.
SP1	When a job is selected for scheduling, this variable contains all the times at which there is a change in either crewmember schedules, resource usage, or time or target constraints.
SPA	When a job is selected for scheduling, this matrix (one column for each crewmember) maintains information as to what each crewmember is doing at each of the times specified in SP1.
AUDX	Contains the audio levels of each of the jobs.
RES	The resource limits of the four resources.
RESM	Contains the resource levels of the of the four jobs for the four resources. This matrix has JL rows and 4 columns.
RESX	For all 4 resources, this variable contains the difference between the resourse level of each job and the resource limit. This matrix has JL rows and 4 columns.
AJS	1 if AUTO JOB SELECT is turned on, 0 otherwise.
ACS	1 if AUTO CREW SELECT is turned on, 0 otherwise.
HEU	The number of the heuristic currently be used for job assignment.
RE	The rating of each job on the Greatest Resourse Usage Heuristic
RE2	The rating of each job on the Greatest Remaining Resourse Usage Heuristic
OC	The parameter information from SET PARAMETERS
MODE	Maintains the current mouse mode.
SS	The current starting point of the timeline which is displayed on the screen.
PIX	The number of minutes per pixel for the timeline displayed on the screen.

C.2.4 Functions for Performing Scheduling

Jobs are selected for scheduling either manually, using the functions JOBC and IR2A (described in Section C.2.8), or by MFIVE, using the function JOBRANK. JOBRANK uses the selected heuristic to find the most highly rated unscheduled pending job. When using AUTO JOB SELECT option on the AUTOPLAN menu, the function AJOB is called. When using the SELECT JOB option on the AUTOPLAN menu, the function AJOB2 is called. AJOB and AJOB2 both utilize JOBRANK to select the job to be scheduled.

Once a job is selected for scheduling, the screen is updated to display parameters relevant to that job using the function SETSCHED. SETSCHED utilizes the function CONPROC, which applies the time, target, and resource constraints to the crewmember timelines to determine which regions are infeasible for the scheduling of the job.

Manual crewmember assignment is performed within the function INIT. If a job requires multiple crewmembers, then the function SELWINDOW asks the user to select from the other available crewmembers. SELWINDOW has two arguments, C and E. C contains the regions on the screen (one for each crewmember) to be clicked on in order to assign the other crewmembers. E contains the total number of crewmembers to be selected. The function returns a 0-1 vector containing a 1 for each crewmember selected, and a 0 for each crewmember not selected.

The function SELWINDOW2 operates similarly to SELWINDOW. SELWINDOW2 is called when the user clicks on the crewmember numbers of crewmembers, in order to assign them the job at their earliest mutually available start time. SELWINDOW2 also requires two arguments, C and E. C contains the number of the first crewmember selected. E contains the total number of crewmembers to be selected. The function returns a vector containing the numbers of the crewmembers selected.

The function CSCHED is utilized by MFIVE to perform automatic crewmember selection. When scheduling jobs requiring multiple crewmembers, CSCHED utilizes the function ISEC to find the earliest mutual start time among groups of crewmembers. ISEC has two arguments VAL and SPB. The variable VAL contains the performance times of each of the crewmembers, and the variable SPB contains a representation of whether or not each crewmember is available at times throughout the planning interval. ISEC returns an explicit result which is the earliest mutual start time of the crewmembers, if one exists. If no mutual feasible start time exists, then the functions returns the time of the end of the flight plan.

Once crewmembers are selected (either manually or automatically) to perform a job, the function ASSIGN is used to update MFIVE's internal representation of the schedule and to propagate the time constraint-linked effects of the job on other jobs. ASSIGN requires two arguments, L and G6. The numeric vector L contains the numbers of the crewmembers chosen to perform the job. The first element of G6 is the chosen start time of the job. The rest of the elements of G6 are the completion times of the all the crewmembers, were they to perform the job at the start time indicated by the first element of G6.

The function AUTOPLAN2 is used to perform autonomous scheduling using the COMPLETE SCHEDULE option on the AUTOPLAN menu. The function AUTOPLAN2 is called from the function AUTOPLAN1, which is called from the function AUTOPLAN. AUTOPLAN and AUTOPLAN1 are "shielding functions": their only purpose is to allow the declaration of more local variables than could comfortably be included in the header of AUTOPLAN2. AUTOPLAN2 utilizes the functions JOBRANK, CONPROC, and CSCHED in performing scheduling. If the Maximum Compatibility Method or the Constrained Maximum Compatibility Heuristic are being used, AUTOPLAN2 uses the function MAT to generate the compatibility matrix. AUTOPLAN2 also uses the function ATASGN2 to simultaneously perform start time and crewmember assignment to many different jobs. ATASGN2 has two arguments, ASM and ASMT. ASM is a 0-1 matrix. If the entry on row x, column y of the matrix is a 1, then it indicates that Crewmember y is to be assigned to Job x. The vector ASMT contains the start times of jobs which are to be assigned by ATASGN2. Negative entries in ASMT indicate jobs that are not be be assigned crewmembers or start times. ATASGN2 returns and explicit value containing the timeline with all the indicated jobs scheduled.

The function STATWINDOW is used take action when the user clicks the mouse inside one of the crewmember's scheduling areas. The specific action taken by STATWINDOW depends on the current setting of the MOUSE MODE menu. STATWINDOW is a function of a single argument, A, which indicates the number of the crewmember whose scheduling area has been clicked in.

The function PRS is used to draw the schedules and resource usages on the Macintosh screen. PRS is a function of a single argument L, which is a numeric vector of length one or two. If the first element of L is equal to negative one, then only the resource usages (the four resources and the audio levels) are updated. If the first element of L is one, then PRS will not erase the schedules on the screen before updating them. If the first element of L is zero, then the scheduling areas will be completely erased before they are redrawn. If the second element of L exists and is one, then the resource usages will be updated. If the second element of L does not exist or is equal to zero, then the resource usages will not be drawn.

C.2.5 APL Provided Functions

The following function are provided with the APL*PLUS system and provide an interface with the Macintosh Toolbox. Details of their operation is provided in [STSC, 1986].

BUTTON	Indicates whether or not the mouse button is depressed.
CLIPRECT	Sets a boundary for drawing.
CLOSEPICTURE	Returns a QuickDraw encoded list of all drawing and text commands.
CUTPICTURE	Returns a QuickDraw picture of the specified portion of the screen.
DELETEMENU	Removes the specified menu number from the menu bar.
DRAWLINE	Draws a line between two specified locations.
DRAWMENUBAR	Redraws the menu bar.
DRAWPICTURE	Draws a picture in the specified portion of the screen.
DRAWTEXT	Draws text at a specified location
ERASERECT	Fills the inside of the specified rectangle with the background pattern.
FILLRECT	Fills the inside of the specified rectangle with the specified pattern.
FRAMERECT	Draws the outline of the specified rectangle.
FRAMEROUNDRECT	Draws the outline of the specified rounded rectangle.
GETMOUSE	Returns the pixel position of the mouse on the screen.
GETPEN	Returns the pixel position of the pen.
HIDECURSOR	Makes the cursor invisible.
INITCURSOR	Sets the cursor to the standard mouse cursor and makes it visible.
INVERTRECT	Inverts all the pixels inside the specified rectangle.
LINETO	Draws a line from the current pen location to the specified spot.
MOVETO	Moves the pen from the current pen location to the specified spot.
OPENPICTURE	Begins storing an encoded list of all drawing and text commands.
PENMODE	Sets the pen mode, which determines graphics overlap.
PENNORMAL	Sets the pen size, mode, and pattern to original values.

PENPAT	Changes the current pen pattern.
PICTFRAME	Returns the original frame size of a QuickDraw picture.
PTINRECT	Indicates whether a point is inside a specified rectangle.
SCROLLRECT	Moves the specified rectangle the specified number of pixels.
SETMENU	Puts a new menu on the menu bar.
SHOWCURSOR	Makes the cursor visible.
STANDMENUBAR	Returns the menu bar to the standard APL*PLUS menu configuration.
TEXTFACE	Specifies the style of subsequent text.
TEXTFONT	Specifies the font of subsequent text.
TEXTMODE	Sets the text drawing mode, which determines how text overlaps.
TEXTSIZE	Specifies the point size of subsequent text.
TEXTWIDTH	Measures the width of a specified character vector in pixels.

C.2.6 Data Manipulation Functions

The following functions all perform manipulation on character data:

Function Name	Arguments	Description
ADJOIN2	A, B	Returns a matrix containing A and B adjoined to each other side by side. A and B must be matrices, with A having at least as many rows as B.
APP		
APPEND	A, B	Returns a matrix containing A and B appended with A above B. A and B can be character scalars, vectors, or matrices.
BL	S	Takes a character vector S and removes any multiple occurrences of the space character from the vector.
CAPPEND	A, B	Same as APPEND, but centers the B matrix beneath the A matrix.

CHARMAT	B, V	Takes the character vector V and changes it into a matrix where the characters in the vector B are taken as delimiters signalling the start of a new line.
CLIST	A, B	Draws the character vector or matrix A at the coordinates indicated by B.
CONVT3	DEF, DATE	Attempts to interpret the character vector DATE as a day and time in the form {Day/Hour:Minute}. If successful, the function will return the number of minutes since time zero. Uses the number specified in DEF to provided defaults for the day and hour.
CVTTD	T	Takes a number T, which specifies a number of minutes since time 0, and converts it to a character vector in the form Days/Hours:Minutes.
CVTTIME1	DEF, T	Inverse of the function CONVT. Takes a number T, which specifies a number of minutes since January 1, 1900 0:00, and converts it to a character vector in the form Month Day, Year Hour:Minute. If T is equal to DEF it is converted to the vector **NO INFO**.
DLBS	A	Removes the first blank space, and everything following it, from the character vector A.
DLBS2	A	Removes all trailing blank spaces from the character vector A.
DLBS3	A	Removes all trailing blank columns from the character matrix A.
DRAWTIME	A, B	Takes B, expressed as a number of minutes, and converts it to the form Hours:Minutes, and then displays the result at the coordinates specified in 2 element vector A.

ELIM	A, B	If A is a vector, eliminates the characters or numbers in the B position(s) of the vector A. If A is a matrix, then the rows indicated by B are eliminated.
FIND	A, B	A and B are character matrices. Returns a 0-1 vector containing a 1 for each row in B which has an exact match in one of the rows in A (trailing spaces are added as necessary). Contains a 0 of each row in B which does not have an exact match in A.
FIND3	W, T	W is a vector character string, T is a character matrix. Returns the row numbers in T in which the character string W appears.
HCEN	A, B	Prints the number A centered horizontally about the coordinates specified in B.
UPPERCASE	A	Changes all lowercase letters in the character vector A to uppercase letters.
VCEN	A, B	Prints the number A centered vertically about the coordinates specified in B.
VCEN2	A	Changes the number A to a character string to be printed out with a width of 8 characters.

C.2.7 The Input Interface Functions

The function GETINPUT in the SCHED workspace is a reduced version of the function GETINP in the PRIME workspace (Section C.1.7). The arguments of the function GETINPUT is identical, except that it only supports the options B[1] = 1 (character input), B[1] = 4 (date and time input), and B[1] = 8 (day and time input).

The functions REPLYWINDOW, ERRORWINDOW, and HELPWINDOW all operate exactly as described in Section C.1.7.

The function BAWINDOW is a dyadic function for specifying whether one job is supposed to precede or follow another job when setting a precedent constraint. BAWINDOW displays the words "BEFORE", "AFTER", and "CANCEL" and waits for the user to click on one of these words in response. BAWINDOW has 2 arguments, C and A. A window is created at the top of the screen, where the message contained in the character vector A is displayed. After the user clicks on one of the options, the window is removed, and the screen restored to its original condition. If the number stored in C is equal to zero, then no precedence constraint is allowed, and an error message is produced (with the function ERRORWINDOW). If C is equal to 1, then no "BEFORE" constraint is allowed, and the word BEFORE is not displayed. If C is equal to 2, then no "AFTER" constraint is allowed, and the word AFTER is not displayed. BAWINDOW returns an explicit result of 0 if no precedence constraint is set, 1 if BEFORE is selected, and 2 if AFTER is selected.

C.2.8 Other User Interface Functions

The dyadic function CH2 allows the user to select off of a list like the one that is displayed when one tries to assign target constraints (see Figure 6.45). The function CH2 is called from the function CHOICE2. CHOICE2 is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of CH2. CH2 has two input arguments, G1 and A. The character matrix G1 contains the list that is to be selected from. The numeric vector A contains the numbers of entries from the list that are not allowed to be selected. Additionally, the character vector MSG is implicitly passed to CH2 from the calling function and specifies the name of the window that is being opened. The function CH2 returns an explicit result which is the row numbers of the item on the list which has been selected.

The niladic function PRIORITYGET is used to provide the window necessary to set the job priorities (Figure 6.39). PRIORITYGET utilizes the functions EDA, EDITA, and ED2A, as which are identical with the functions with the same name that were described in Section C.1.8.

The monadic function RESEN allows the user to generate an enlarged picture of one of the resource constraints (see Figure 6.26). The function RESEN is called from the function RESENL. RESENL is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of RESEN. The single argument of RESEN is the variable R, which indicates which resource picture is to be enlarged. R = 1 for Power Usage, R = 2 for Transmission Usage, R = 3 for Memory Usage, and R = 4 for Multiplex Usage. RESEN uses the function GRAPH2 to draw the resource usage graph. GRAPH2 has two arguments T and H. T is a seven element numeric vector. T[1] specifies the start of the current timeline, T[2] specifies the end of the current timeline, T[3] specifies the maximum resource usage, and the last four elements of T specify the size of the window in which the graph is to be drawn. H is a two column matrix. The first column contains resource usages and the second column contains time markers corresponding to those resource usages.

The function JOBC is used to display the Job Information window (see Figure 6.28). JOBC has a single argument, L. If L is equal to 12, then the information window is opened (if it was closed) or is closed (if it was open). Any other value of L indicates a regions within the information window that has been clicked on with the mouse. Whenever the mouse is clicked on one of the job names inside of the window (thus selecting or deselecting a job for scheduling), the function IR2A is executed. IR2A is a niladic function which inverts the appropriate areas within the Job Information window and prepares the main screen (by displaying the number of crewmembers required to perform the job, drawing the scheduling bars for each of the individual crewmembers, blacking out infeasible scheduling intervals, etc.).

The niladic function SETCON allows the user to enter the time constraints (see Figure 6.40). The function SETCON is called from the function SETCON2, which is called from the function SETCON1. SETCON1 and SETCON2 are "shielding functions": their only purpose is to allow the declaration of more local variables than could comfortably be included in the header of SETCON. Clicking on the names of one of the jobs in the time constraints window will execute the niladic function IRCON, which updates the time constraint window to display all constraints relating to the job selected (see Figure 6.42). Clicking to add a time constraint causes the execution of the niladic function IRC. IRC and IRCON utilize the functions TCEST, TCLST, TCMAX and TCMIN. All the functions have a single argument, A. TCEST returns a vector containing the earliest start times for the jobs indicates in A. TCLST returns a vector containing the latest start times for the jobs indicated in A. TCMAX returns a matrix, each row of which contains the maximum allowable difference in start times between the start of each job in A and the start of each of the other jobs. TCMIN returns a matrix, each row of which contains the minimum allowable difference in start times between the start of each job in A and the start of each of the other jobs.

The function DRAWMSG is used by all of the above window producing functions to write the name of the window in the bar at the window's top. DRAWMSG prevents the window name from running outside the window (by truncation) if it is longer than can fit inside the bar. DRAWMSG has two arguments, A and B. The character vector A contains the name of the window that is to be writen. B is a 2 element numeric vector. B[1] contains the number of pixels which will be needed to (fully) draw character vector A, and B[2] contains the number of pixels actually available to draw the name.

The monadic function SETTAR is used to provide the window necessary to add and revise target constraints (Figure 6.46). SETTAR has a single argument J, which indicates the number of the job to which target constraints are to be added. SETTAR returns an explict value of 1 if the user clicks on CANCEL. Otherwise a value of 0 is returned.

The niladic function SETPIX is used to provide the window necessary to explicitly change the timeline scale and origin (Figure 6.25). SETPIX returns an explicit value of 1 if the user clicks on CANCEL. Otherwise a value of 0 is returned.

The niladic function SETSEARCH is used to provide the window necessary to set the search parameters (Figure 6.37). SETSEARCH returns an explicit value of 1 if the user clicks on CANCEL. Otherwise a value of 0 is returned.

The monadic function JOBATT is used to provide the window necessary to modify job start time and assignment using the RESCHEDULE NUMERICALLY option on the MOUSE MODE menu (see Figure 6.49). The function JOBATT is called from the function JOBATT1. JOBATT1 is a "shielding function": its only purpose is to allow the declaration of more local variables than could comfortably be included in the header of JOBATT. JOBATT has a single argument H, which is the number of the job which has been clicked on for numerical rescheduling.

The function PRTSCHED is used when the user selects the PRINT SCHEDULE option from the AUTOPLAN menu. PRTSCHED draws a window which allows the user to specify parameters of desired output. PRTSCHED then calls the function PRTSCHED2 which produces the actual output.

C.2.9 Data Transfer from the PRIME Workspace

When data is transferred from the PRIME workspace, a data file called "MAN" is placed in the same folder as the SCHED workspace. The file MAN contains a prefix which signifies the name of a file in which the transferred data is to be found. This prefix always consists of the word "MAN" followed by an integer. Assume, for example, that the file MAN contains the prefix "MAN2" The data in the file MAN2 will then be loaded into the SCHED workspace by the function EF (called from the function START, which was discussed in Section C.2.1). EF has a single argument, A, which is a character string equal to the name of this file ("MAN2"). Each variable stored in the file MAN2 will be given the prefix "MAN2". For example, the variable in the file labelled PD will be stored in the variable MAN2PD. Finally, the function EF will create a variable (called MAN2VARS) which contains the names of all the variables which have been loaded into memory.

Once these variables are loaded into memory, the function START calls the function S, which in turn calls the function INIT. INIT then calls the function SETUP, and SETUP calls the function SETJOB. The functions SETUP and SETJOB transform the information passed from the PRIME workspace into the format required by the scheduler. In doing so, these functions initialize most of the variables discussed in Section C.2.3. SETUP also has the task of setting up the menus used by the schedule, while SETJOB sets up the information seen inside the Job Information window. SETUP is a niladic function, while SETJOB requires a single argument R, which contains the name of the prefix of the variables which were transferred from the PRIME workspace.

C.2.10 File Manipulation Functions

The function FILEOPEN is a monadic function which is used to get from the user (via a dialog box) the name of a file (containing a database) which is to be loaded into MFIVE. The function has a single argument, P, which is a character vector containing the prefix (ther first letters) which the file name must possess. The function returns an explicit result, which is the name of the file to be opened.

The function DATALOAD is a monadic function which load the database into MFIVE. The function has a single argument, WO, which is the name of the file to be loaded. The function has no explicit result.

The function FILESAVE is a dyadic function which is used to get from the user (via a dialog box) the name of a file a MFIVE database is to be saved. The function has two arguments, A and B. The variable B contains a character vector containing the prefix (the first letters) which the file name must possess. The variable A contains a message displayed to the user in the dialog box. The function returns an explicit result, which is the name of the file to be opened.

The function DATASAVE is a dyadic function which saves an MFIVE database. The function has two arguments, W7 and KK. W7 is a character vector which contains the name of the file in which the database is to be saved. KK is a character matrix containing the names of the variables to be saved in the file. The function returns an explicit result, which is the total number of variables saved (rows in the KK matrix) plus 1.

C.2.11 Miscellaneous Functions

The function TEXTNORMAL returns the textfont, textsize, and textface to a preset arrangement.

The function T resets text parameters, the cursor, pen, and menu bar to a preset arrangement. It also unties any tied files.

The function CLEAR draws the word MFIVE in the bar at the top of the screen, and defines the size of the screen used by MFIVE.

The function CLEARBUFFER clears any key or mouse entries queued in the input buffer.

The function DELETEMENUS deletes the OPTIONS, AUTOPLAN, CONSTRAINTS, and MOUSE MODE menus. The function PUTMENUS replaces these menus.

C.2.12 Functions External to the Scheduler

The following functions are not directly used by the scheduler, but located in the SCHED workspace:

The function WS returns the name of the Macintosh folder in which the SCHED folder (and presumably the PRIME folder) are located. The function PRIME loads the PRIME workspace.

The function HCL is used to produce screen or printer listing of APL functions. HCL has a single argument, INS. INS is a character vector which contains the names of the functions to be listed, separated by spaces. By specifying INS to be the character string "ALL", all of the functions in the workspace will be listed, followed by a listing of all variable names.

The STATS function produces summary statistics for a scheduling problem of the type found in Appendix A. In order to use this function, the user should first load the scheduling problem of interest into MFIVE. Next, select the STOP option from the STOP menu, which will return the user to the APL environment, with MFIVE "suspended." Typing the word STATS, followed by a carriage return, will execute the function STATS and produce the summary statistics for the scheduling problem. Lastly, to return to the scheduler, type "]32" followed by hitting the return key. (The] character should appear on the Macintosh Plus screen as a right pointing arrow.) Selecting the REDRAW SCREEN option from the OPTIONS menu will return the screen to normal.

The function STATS utilizes the functions ATASGN3, CUR, RESPRT, and RESPRT2. ATASGN3 is used to produce an "early start time schedule," where each job is assigned to start at its earliest start time (from the time constraints), without regard to the resource constraints. Generating this schedule is necessary in order to evaluate many of the summary statistics. The function CUR is used to place the cursor at a desired point on the screen. CUR has a single argument, A, which indicates the number of character spaces from the left margin that the cursor should move. RESPRT and RESPRT2 are used to format output of some of the statistics. Both functions have two arguments, B and A. B is a character vector containing the name of the statistic being printed. A contains the values of the statistic for each of the resources. RESPRT lists the values of the statistic for each of the resources, followed by the average and variance values. RESPRT2 also lists the values of the statistics, followed by the total value for all the resources.

The function IRAUTO takes the time constraints encoded in the vector AUTV and applies them all at once without the need to add them individually through the time constraints window.

The function MULT is used to reduce or increase the number of jobs in a scheduling problem. It can be accessed in the same way as described for the function STATS. MULT has a single argument K, which signifies the number of jobs to which the scheduling problem is to be scaled. For example, if an initial problem has 19 jobs, and MULT is executed, setting K to 5 (e.g., by typing "MULT 5" followed by hitting the return key), then a new scheduling problem will be created containing only the first 5 jobs in the original problem. If MULT is executed again, setting K to 30, then another scheduling problem will be created containing 6 copies of each of these 5 jobs. MULT only works accurately on schedules without time constraints, and should be executed when all jobs are unscheduled.

C.2.13 Function Listings for the SCHED Workspace

```
      )FNS
ADJOIN2        AJOB           AJOB2          APPEND         ASSIGN
ATASGN2        ATASGN3        AUTOPLAN       AUTOPLAN1      AUTOPLAN2
BAWINDOW       BL             BUTTON         CAPPEND        CH2
CHARMAT        CHOICE2        CLEAR          CLEARBUFFER    CLIPRECT
CLIST          CLOSEPICTURE   CONPROC        CONVT3         CSCHED
CUR            CUTPICTURE     CVTID          CVTTIME1       DATALOAD
DATASAVE       DELETEMENU     DELETEMENUS    DLBS           DLBS2
DLBS3          DRAWLINE       DRAWMENUBAR    DRAWMSG        DRAWPICTURE
DRAWTEXT       DRAWTIME       ED2A           EDA            EDITA
EF             ELIM           ERASERECT      ERRORWINDOW    FILEOPEN
FILESAVE       FILLRECT       FIND           FIND3          FRAMERECT
FRAMEROUNDRECT GETINPUT       GETMOUSE       GETPEN         GRAPH2
HCEN           HCL            HELPWINDOW     HIDECURSOR     INIT
INIT1          INIT2          INIT3          INIT4          INITCURSOR
INVERTRECT     IR2A           IRAUTO         IRC            IRCON
ISEC           JOBATT         JOBATT1        JOBC           JOBRANK
LINETO         MAT            MOVETO         MULT           OPENPICTURE
PENMODE        PENNORMAL      PENPAT         PICTFRAME      PRIME
PRIORITYGET    PRS            PRTSCHED       PRTSCHED2      PTINRECT
PUTMENUS       REPLYWINDOW    RESEN          RESENL         RESPRT
RESPRT2        S              SCROLLRECT     SELWINDOW      SELWINDOW2
SETCON         SETCON1        SETCON2        SETJOB         SETMENU
SETPIX         SETSCHED       SETSEARCH      SETTAR         SETUP
SHOWCURSOR     SL             STANDMENUBAR   START          STATS
STATWINDOW     T              TCEST          TCLST          TCMAX
TCMIN          TEXTFACE       TEXTFONT       TEXTMODE       TEXTNORMAL
TEXTSIZE       TEXTWIDTH      UPPERCASE      VCEN           VCEN2
WS
```

```
      ∇R←A ADJOIN2 B
[1]    R←A,((ρA)[ONE],(ρB)[TWO])↑B
      ∇
```

```
      ∇AJOB;G2;JA;JWT;HEUL
[1]    JWT←PRI⋄JA←IZ⋄HEUL←HEU+TEN×OC[SIX]=TWO⋄→(ZERO≠ρ,JS)/B7
[2]    B1:→(HEU∊9 10)/B9⋄G2←⎕GETKEY⋄→(ZERO≠ρG2)/B5⋄JS←JOBRANK⋄→(ZERO=ρJS)/B2
[3]    B7:OPENPICTURE SCREEN⋄SETSCHED
[4]    →(ZERO=ρJS)/B1⋄→ACS/B6⋄→ZERO
[5]    B2:→(~ACS)/B3⋄OPENPICTURE SCREEN⋄PRS NONE⋄SCREEN DRAWPICTURE CLOSEPICTURE
[6]    B3:→(ZERO≠ρJA)/B8
[7]    ERRORWINDOW 'ALL JOBS HAVE BEEN SCHEDULED.  SCHEDULE IS COMPLETE'
[8]    B4:AJS←ZERO⋄ACS←ZERO⋄PLANMENU[2 4;TWO]←' '⋄PLANN SETMENU PLANMENU⋄→ZERO
[9]    B5:→(~G2≡TWO,PLANN,ONE)/B1
[10]   OPENPICTURE SCREEN⋄PRS NONE⋄SCREEN DRAWPICTURE CLOSEPICTURE⋄→B4
[11]   B6:JA←JA,JS⋄ERASERECT 3 4ρ207 370 218 510 25 438 33 510 32 378 168 510
[12]   SPA←SP⋄ERASERECT BR⋄GRAYPAT FILLRECT M4⋄FRAMERECT M4⋄BR←0 4ρIZ⋄JS←IZ
[13]   RECT[19+ICL;]←ZERO⋄→B1
[14]   B8:G2←'SCHEDULING OF ALL JOBS HAS BEEN ATTEMPTED.  ',(⍕ρJA),' JOB'
[15]   ERRORWINDOW G2,((ONE≠ρJA)↑'S'),' REMAIN',((ONE=ρJA)↑'S'),' UNSCHEDULED'
[16]   →B4
[17]   B9:G2←'AUTO JOB SELECT CANNOT BE USED WITH THE SELECTED HEURISTIC'
[18]   ERRORWINDOW G2⋄→B4
```

```
∇

∇AJOB2;G2;JA;JWT;HEUL
 →(HEU∊9 10)/B4
 JWT←PRI◊JA←IZ◊HEUL←HEU+TEN×OC[SIX]=TWO◊JS←JOBRANK◊→(ZERO=ρJS)/B2
 OPENPICTURE SCREEN◊SETSCHED◊→(~ACS)/ZERO◊OPENPICTURE SCREEN◊PRS NONE
 SCREEN DRAWPICTURE CLOSEPICTURE◊→ZERO
 B2:→(~ACS)/B3◊OPENPICTURE SCREEN◊PRS NONE◊SCREEN DRAWPICTURE CLOSEPICTURE
 B3:ERRORWINDOW 'ALL JOBS HAVE BEEN SCHEDULED.  SCHEDULE IS COMPLETE'
 ACS←ZERO◊PLANMENU[FOUR;TWO]←' '◊→ZERO
 B4:ERRORWINDOW 'SELECT JOB CANNOT BE USED WITH THE SELECTED HEURISTIC'
∇


∇R←A APPEND B;D;D1;D2
 D1←NTWO↑ONE,ρA◊D2←NTWO↑ONE,ρB◊A←D1ρA◊B←D2ρB◊D←NONE↑D1⌈D2
 R←((D1[ONE],D)↑A)⍪(D2[ONE],D)↑B
∇


∇L ASSIGN G6;K;KK;G2;H;P;J;N
 G2←G6[ONE]◊G6←ONE↓G6◊H←G6[L]◊SP1←SP[;CL6]◊K←G2,((⍳ρH)=H⍳H)/H◊K←(~K∊SP1)/K
 →(ZERO=ρK)/B1◊KK←SP1,K◊SP1←KK[⍋KK]◊SP←SP[+\~SP1∊K;]◊SP[;CL6]←SP1
 B1:K←SP1⍳G2◊→(ZERO=CS)/B10◊P←K◊J←H
 B2:KK←⌊/J◊N←SP1⍳KK◊SP[NONE+K+⍳N-K;(H≥KK)/L]←JS◊J←(J>KK)/J◊K←N
 →(ZERO≠ρJ)/B2◊N←N-P◊J←NONE+P+⍳N
 B11:SP[J;CL4]←SP[J;CL4]+(N,FOUR)ρRESM[JS;]
 SP[J;CL5]←×SP[J;CL5]+AUDX[JS]◊P1[JS;L]←ONE◊ST[JS]←G2◊WL[L]←WL[L]+PDS[L]
 ST1[JS]←KK◊JSS←JSS,JS◊→AP/B3◊RECT[19+ICL;]←ZERO◊OPENPICTURE SCREEN
 MOVETO 176 460◊TEXTFACE ONE◊DRAWTEXT EIGHT↑VCEN2 +/PRI[JSS]
 TEXTFACE ZERO◊ERASERECT 3 4ρ207 370 218 510 25 443 33 510 32 378 168 510
 →(ACS∧AJS)/B9◊PRS NONE
 B9:ERASERECT BR◊GRAYPAT FILLRECT M4◊FRAMERECT M4
 SCREEN DRAWPICTURE CLOSEPICTURE◊BR←0 4ρIZ
 B3:J←JS+ONE◊MPD[JS]←KK-G2◊MPE[JS]←KK-G2
 K←ONE-(PRC⍳JS)↓(NONE+PRC⍳J)↑PRC
 B5:→(ZERO=ρK)/B6◊P←K[ONE]◊CONP[P;J]←G2-KK
 CONP←CONP⌊CONP[;P]∘.+CONP[P;]◊K←ONE↓K◊→B5
 B6:CONP[ONE;J]←G2◊CONP[J;ONE]←-G2◊T3[JS]←KK
 CONP←CONP⌊CONP[;ONE]∘.+CONP[ONE;]◊CONP←CONP⌊CONP[;J]∘.+CONP[J;]
 T2←CONP[ONE;ONE+IJL]◊T3←T3⌊T2+MPE◊JS←IZ◊K←ONE
 B4:T3[K]←⌊/T3[K],T2[-(PRC⍳K)↓(NONE+PRC⍳K+ONE)↑PRC]◊K←K+ONE◊→(K≤JL)/B4
 →AP/ZERO◊WTEXT[J+TWO;ONE]←'*'
 →TW/ZERO◊SW←CUTPICTURE B◊→AJS/ZERO◊B DRAWPICTURE JW
 K←NONE↑JSS◊→(TC≥K+THREE)/ZERO◊→(BC<K+THREE-TC)/ZERO
 →(LC>ZERO)/B7◊ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]
 OUT←(BC,RC)↑(TC,LC)↓WTEXT◊MOVETO 10 8+S1,S2◊DRAWTEXT OUT◊→B8
 B10:N←(SP1⍳H)-K◊J←NONE+K+⍳N◊KK←H◊L←IZ◊→B11
 B7:K←11×THREE+K-TC◊INVERTRECT D+K,ZERO,K,SIX×RC
 B8:JW←CUTPICTURE B
∇


∇SP←ASM ATASGN2 ASMT;K;KK;G2;E;H;P;J;N;JS;L;SP1;M
 JS←ONE◊SP←(TWO,CL6)ρZERO◊M←ASMT≥ZERO◊SP1←IZ
 B14:→(~M[JS])/B13◊SP1←SP1,ASMT[JS]+PD[JS;((JT[JS]=ZERO)ρONE),ASM[JS;]/ICL]
 B13:JS←JS+ONE◊→(JS≤JL)/B14◊JS←ONE◊SP1←SP1,ZERO,MD,M/ASMT
 SP1←((⍳ρSP1)=SP1⍳SP1)/SP1◊SP1←SP1[⍋SP1]◊SP←((ρSP1)[ONE],CL6)ρZERO
 B11:L←ASM[JS;]/ICL◊→(~M[JS])/B10◊G2←ASMT[JS]
```

```
    H←G2+PD[JS;L]
    B1:K←SP1⍳G2◇P←K◇J←H
    B2:→(ZERO=⍴L)/B3◇KK←⌊/J◇N←SP1⍳KK◇SP[NONE+K+⍳N-K;(H≥KK)/L]←JS◇J←(J>KK)/J
    K←N◇→(ZERO≠⍴J)/B2
    B7:N←N-P◇J←NONE+P+⍳N
    SP[J;CL4]←SP[J;CL4]+(N,FOUR)⍴RESM[JS;]◇SP[J;CL5]←×SP[J;CL5]+AUDX[JS]
    WL[L]←WL[L]+H-G2◇J←JS+ONE
    K←ONE-(PRC⍳JS)↓(NONE+PRC⍳J)↑PRC
    B5:→(ZERO=⍴K)/B6◇P←K[ONE]◇CONP[P;J]←G2-KK
    CONP←CONP⌊CONP[;P]∘.+CONP[P;]◇K←ONE↓K◇→B5
    B3:KK←G2+PD[JS;ONE]◇N←SP1⍳KK◇→B7
    B6:CONP[ONE;J]←G2◇CONP[J;ONE]←-G2◇T3[JS]←KK
    CONP←CONP⌊CONP[;J]∘.+CONP[J;]
    B10:JS←JS+ONE◇→(JS≤JL)/B11◇SP[;CL6]←SP1
    CONP←CONP⌊CONP[;ONE]∘.+CONP[ONE;]
    T2←CONP[ONE;ONE+⍳JL]◇T3←T3⌊T2+MPE◇K←ONE
    B4:T3[K]←⌊/T3[K],T2[-(PRC⍳K)↓(NONE+PRC⍳K+ONE)↑PRC]◇K←K+ONE◇→(K≤JL)/B4
   ∇


   ∇SP←ATASGN3;K;G2;J;N;JS;SP1;A
    A←ONE↓-CT[;ONE]◇JS←ONE◇SP1←A+MPD◇SP1←SP1,ZERO,MD,A
    SP1←((⍳⍴SP1)=SP1⍳SP1)/SP1◇SP1←SP1[⍋SP1]◇SP←((⍴SP1),SIX)⍴ZERO
    SP[;SIX]←SP1
    B1:G2←A[JS]◇K←SP1⍳G2◇N←(SP1⍳G2+MPD[JS])-K◇J←NONE+K+⍳N
    SP[J;ONE+⍳FOUR]←SP[J;ONE+⍳FOUR]+(N,FOUR)⍴RESM[JS;]◇SP[J;ONE]←SP[J;ONE]+JT[JS]
    JS←JS+ONE◇→(JS≤JL)/B1
   ∇


   ∇AUTOPLAN;Q;W2;ML;MM;KK;R0;R1;R2;R3;R4;R5;R7;R8;HEUL;MA4;R9;JWT;V1;V2;V3;V4
    AUTOPLAN1
   ∇


   ∇AUTOPLAN1;V5;V6;HEUL
    AUTOPLAN2
   ∇


   ∇AUTOPLAN2;H1;H2;UN2;UN;JA;JWT;Q;G1;G2;A;C;D;E;F;G;H;J;N;P;R;V;K;C1;M;W1;DD
    UN←50 50 274 434◇UN2←⎕GETBITS UN◇50 50 272 426 DRAWPICTURE AUTOPICT
    TEXTFACE ONE◇MOVETO 98 190◇DRAWTEXT (ELEVEN×OC[SIX]-ONE)↑'RANDOMIZED'
    MOVETO 112 118◇DRAWTEXT DLBS2 ONE↓HEURISTICS[HEU;]
    MOVETO 132 222◇DRAWTEXT ⍕OC[FOUR]
    MOVETO 144 222◇DRAWTEXT ⍕OC[FIVE]◇MOVETO 156 222◇DRAWTEXT ⍕JL
    MOVETO 222 260◇DRAWTEXT '    0'◇MOVETO 234 260◇DRAWTEXT '    0'
    MOVETO 246 260◇DRAWTEXT '    0'◇MOVETO 258 260◇DRAWTEXT '    0'
    MOVETO 177 170◇DRAWTEXT '    1'◇MOVETO 188 170◇DRAWTEXT '    1'
    MOVETO 200 375◇DRAWTEXT '    0'◇TEXTFACE ZERO
    G1←⎕AI[TWO]+(60000×OC[ONE])+1000000000×OC[ONE]<ZERO◇G2←G1◇R9←ZERO
    A←((TWO,CL5)⍴ZERO),2 1⍴ZERO,MD◇C←JL⍴NONE◇D←C◇E←CT◇F←MPF◇G←MPG
    P←(JL,CL)⍴ZERO◇H←TT◇J←MD-MPF◇N←CL⍴ZERO◇R←IZ◇AP←ONE◇V1←~HEU∊9 10
    V2←(60000+OC[TWO])+1000000000×OC[TWO]<ZERO◇V3←CL⌈EIGHT×JT=ZERO
    V4←OC[THREE]=ONE◇V5←1 0↓0 1↓CT≤ZERO◇JA←IZ◇→V1/B46◇MPD←MPF◇MAT
    B46:AT←(ZERO,SEVEN+JL)⍴ZERO◇W2←ONE◇HEUL←HEU+TEN×OC[SIX]=TWO◇C1←ONE
    DELETEMENUS◇SEARCHN SETMENU SEARCHMENU
    DRAWMENUBAR◇V←TWO,SEARCHN,ONE◇→TW/B2◇B DRAWPICTURE SW
    B2:H1←MD◇H2←TT◇TT←TT⌊MD◇JWT←PRI◇V6←ONE◇M←ZERO◇DD←ZERO
    JSS←NONE◇Q←⎕AI[TWO]◇→V1/B25◇MM←(JL,JL)⍴,MA4◇→B25
```

```
B6:MPD←MPF⋄MPE←MPG⋄CONP←CT⋄T3←TT⋄T2←MD-MPD⋄SP←((TWO,CL5)ρZERO),2 1ρZERO,MD
WL←CLρZERO⋄ST←JLρNONE⋄ST1←ST⋄JSS←IZ⋄P1←(JL,CL)ρZERO
B4:K←⎕GETKEY⋄→(ZERO≠ρK)/B7⋄JS←JOBRANK⋄→(IZ≡JS)/B5
B24:CS←JT[JS]⋄PDS←V3[JS]ρPD[JS;]⋄CONPROC⋄CSCHED⋄TEXTFACE ONE
MOVETO 200 375⋄DRAWTEXT NFIVE↑⍕ρJSS⋄MOVETO 200 170
DRAWTEXT NFIVE↑⍕NONE↑JSS⋄TEXTFACE ZERO⋄→(~V4)/B39⋄→(IZ≡JS)/B4
B37:G3←+/PRI[JSS]
B27:ST[JS]←ONE↑NTWO↑SP1⋄K←JS
B47:K←(ST[K]=ST-ONE↓CT[ONE+K;])/IJL⋄KK←K[ST[K]⍳⌊/ST[K]]
K←V5[KK;]/IJL⋄→(~V1)/B40⋄JWT[K]←JWT[K]×ONE+(?(ρK)ρTEN)÷TEN⋄→B41
B39:→(~IZ≡JS)/B37⋄→V6/B40⋄→(DD>+/ST1[JSS])/B4
B22:K←ST1⍳⌈/ST1⋄→B47
B40:KK←(JL,ρK)ρONE+(?(ρK)ρTEN)÷TEN⋄MM[K;]←MM[K;]×⍉KK⋄MM[;K]←MM[;K]×KK
B41:W2←W2+ONE⋄→(W2>OC[FIVE])/B11
B15:→(M<G3)/B5⋄→(M≠G3)/B33⋄→V4/B32⋄→(DD>+/ST1[JSS])/B5⋄→B33
B32:→(DD>⌈/ST1)/B5
B33:→(⎕AI[TWO]>G1,G2)/B1,B11
B25:TEXTFACE ONE⋄MOVETO 188 170⋄DRAWTEXT NFIVE↑⍕W2⋄MOVETO 200 375
DRAWTEXT '    0'⋄MOVETO 200 170⋄DRAWTEXT '     '
TEXTFACE ZERO⋄→(HEU∊3 8 10)/B6
R8←MPD⋄R1←MPE⋄R2←T2⋄R3←T3⋄R4←ST⋄R5←ST1⋄R7←P1⋄R0←JL↑JSS⋄MPD←MPF
MPE←MPG⋄T3←TT⋄T2←MD-MPD⋄ST←JLρNONE⋄ST1←ST⋄P1←(JL,CL)ρZERO⋄K←ONE⋄JSS←IZ
B23:JS←JOBRANK⋄→(R0[K]≠JS)/B21⋄JSS←JSS,JS⋄K←K+ONE⋄ST1[JS]←R5[JS]
→(K≤JL)/B23⋄→B26
B21:WL←CLρZERO⋄CONP←CT⋄MPD[JSS]←R8[JSS]⋄MPE[JSS]←R1[JSS]⋄T3[JSS]←R3[JSS]
T2[JSS]←R2[JSS]⋄ST[JSS]←R4[JSS]⋄P1[JSS;]←R7[JSS;]
→(ZERO=R0[K])/B27⋄SP←P1 ATASGN2 ST⋄→B24
B5:K←CUTPICTURE 50 50 272 426
UN2 ⎕PUTBITS UN⋄TEXTNORMAL⋄OPENPICTURE SCREEN⋄JS←IZ⋄PRS ZERO⋄V6←JL≠ρJSS
SCREEN DRAWPICTURE CLOSEPICTURE⋄UN2←⎕GETBITS UN
50 50 272 426 DRAWPICTURE K⋄TEXTFACE ONE⋄MOVETO 222 260
DRAWTEXT NFIVE↑⍕ρJSS⋄MOVETO 234 260⋄DRAWTEXT NFIVE↑⍕⌈/ST1⋄MOVETO 246 260
DRAWTEXT NFIVE↑⍕+/ST1[JSS]⋄MOVETO 258 260
DRAWTEXT NFIVE↑⍕⌊PFIVE+(+/WL×WL)*PFIVE⋄TEXTFACE ZERO⋄M←+/PRI[JSS]⋄G3←M
→V4/B35⋄DD←+/ST1[JSS]⋄→(M<R9)/B48⋄→(M≠R9)/B34⋄→(DD≥+/D[JSS])/B48⋄→B34
B35:DD←⌈/ST1⋄→(M<R9)/B48⋄→(M≠R9)/B34⋄→(DD≥⌈/D)/B48
B34:A←SP⋄C←ST⋄D←ST1⋄E←CONP⋄F←MPD⋄G←MPE⋄P←P1⋄H←T3⋄J←T2⋄N←WL⋄R←JSS⋄R9←M
TEXTFACE ONE⋄MOVETO 222 345⋄DRAWTEXT NFIVE↑⍕ρJSS⋄MOVETO 234 345
DRAWTEXT NFIVE↑⍕⌈/ST1⋄MOVETO 246 345⋄DRAWTEXT NFIVE↑⍕+/ST1[JSS]
MOVETO 258 345⋄DRAWTEXT NFIVE↑⍕⌊PFIVE+(+/WL×WL)*PFIVE⋄TEXTFACE ZERO
B48:G2←⎕AI[TWO]+V2
AT←AT⍪W2,(ρJSS),(⌈/ST1),(+/ST1[JSS]),HEUL,(+/WL),(JL↑JSS),⌊PFIVE+(⎕AI[TWO]-Q)÷1000
→(JL≠ρJSS)/B25⋄→(~V4)/B22⋄MD←NONE+⌈/T3⋄TT←TT⌊MD⋄→B22
B26:K←ST1⍳⌈/ST1⋄K←(ST[K]=ST-ONE↓CT[ONE+K;])/IJL⋄KK←K[ST[K]⍳⌊/ST[K]]
K←V5[KK;]/IJL⋄→(~V1)/B44⋄JWT[K]←JWT[K]×ONE+(?(ρK)ρTEN)÷TEN⋄→B45
B44:KK←(JL,ρK)ρONE+(?(ρK)ρTEN)÷TEN⋄MM[K;]←MM[K;]×⍉KK⋄MM[;K]←MM[;K]×KK
B45:W2←W2+ONE⋄ST1←JLρNONE⋄K←ONE⋄JSS←IZ⋄→(W2>OC[FIVE])/B11
→(⎕AI[TWO]>G1,G2)/B1,B11⋄→B23
B11:W2←ONE⋄TT←H2⋄MD←H1⋄→(C1=OC[FOUR])/B1⋄C1←C1+ONE⋄MOVETO 177 170
TEXTFACE ONE⋄DRAWTEXT NFIVE↑⍕C1⋄MOVETO 200 375⋄DRAWTEXT '    0'
MOVETO 200 170
DRAWTEXT '     '⋄MOVETO 222 260⋄DRAWTEXT '    0'⋄MOVETO 234 260
DRAWTEXT '    0'⋄MOVETO 246 260⋄DRAWTEXT '    0'⋄MOVETO 258 260
DRAWTEXT '    0'⋄TEXTFACE ZERO⋄G2←⎕AI[TWO]+V2⋄→B2
B7:→(~V≡K)/B4
B1:→(H1=NONE↑A[;CL6])/B3⋄A←A⍪(CL5ρZERO),H1
B3:SP←A⋄ST←C⋄ST1←D⋄CONP←E⋄MPD←F⋄MPE←G⋄P1←P⋄T3←H⋄T2←J⋄WL←N⋄TT←H2⋄JS←IZ
JSS←R⋄MD←H1⋄TEXTNORMAL⋄UN2 ⎕PUTBITS UN⋄OPENPICTURE SCREEN⋄PRS ZERO
SCREEN DRAWPICTURE CLOSEPICTURE⋄WTEXT[THREE+JSS;ONE]←'*'⋄PUTMENUS
AP←ZERO⋄→TW/ZERO⋄JOBC 11⋄→ZERO
∇
```

```
     ∇R←C BAWINDOW A;OS;K;M;D;UN;UN2;B;RECT;G1
[1]   R←ZERO◇D←22 20 70 480◇UN←22 20 70 484◇RECT←1 4⍴35 398 56 472◇B←IZ
[2]   →(C≠ZERO)/B8◇ERRORWINDOW 'No Precedence Constraint Is Allowed Here'◇→ZERO
[3]   B8:UN2←⎕GETBITS UN◇ERASERECT D◇FRAMERECT 2 4⍴D,24 22 68 478
[4]   FRAMEROUNDRECT 2 6⍴37 400 54 470 15 15 35 398 56 472 17 17
[5]   TEXTFACE 65◇MOVETO 49 412◇DRAWTEXT 'CANCEL'◇→(C=ONE)/B6◇B←B3
[6]   FRAMEROUNDRECT 2 6⍴37 240 54 310 15 15 35 238 56 312 17 17
[7]   MOVETO 49 252◇DRAWTEXT 'BEFORE'◇RECT←RECT⍪35 238 56 312
[8]   B6:→(C=TWO)/B5◇FRAMEROUNDRECT 2 6⍴37 320 54 390 15 15 35 318 56 392 17 17
[9]   MOVETO 49 337◇DRAWTEXT 'AFTER'◇RECT←RECT⍪35 318 56 392◇B←B,B4
[10]  B5:B←B7,B◇TEXTFACE ONE◇K←35+FIVE×FOUR-THREE⌊⌈(⍴A)+25
[11]  B1:MOVETO K,50◇OS←25-(⌽25↑A)⍳' '◇DRAWTEXT OS↑A
[12]  A←(ONE+OS)↓A◇K←K+ELEVEN◇→(ZERO≠⍴A)/B1◇TEXTFACE ZERO◇G1←⍳ONE↑⍴RECT
[13]  B2:M←⎕GETKEY◇→(THREE≠⍴M)/B2◇→(TWO=M[ONE])/B2
[14]  L←(M[2 3] PTINRECT RECT)/B◇→(ZERO=⍴L)/B2◇→L
[15]  B3:R←ONE◇→B7
[16]  B4:R←TWO
[17]  B7:→(ZERO≠⍴⎕GETKEY)/B7◇UN2 ⎕PUTBITS UN◇→ZERO
     ∇


     ∇R←BL S;C
[1]   C←S≠' '◇R←(C∨ONE⌽C)/S
     ∇


   ∇BUTTON∇


     ∇R←A CAPPEND B;D;E;F;G
[1]   E←NTWO↑ONE,⍴A◇A←E⍴A◇F←NTWO↑ONE,⍴B◇B←F⍴B◇→(ZERO≠(⍴B)[ONE])/B1◇B←B⍪' '◇F←⍴B
[2]   B1:D←NONE↑E⌈F◇G←PFIVE×NONE↑F-E
[3]   R←((ZERO⌊-⌊G)⌽(E[ONE],D)↑A)⍪(ZERO⌊⌈G)⌽(F[ONE],D)↑B
     ∇


     ∇S←G1 CH2 A;S1;S2;S3;S4;OUT;LC;ULC;B;RECT;MS;CR;TR;UN;UN2;C1;JL;G2;G3;G4;G5
[1]   N←18◇OUT←18 0⍴IZ◇K←ZERO◇JL←(⍴G1)[ONE]
[2]   B31:OUT←OUT ADJOIN2 (DLBS3 G1[K+⍳N⌊JL-K;]),AV125◇K←K+N◇→(JL>K)/B31
[3]   FL←NONE,(OUT[ONE;]=AV125)/⍳(⍴OUT)[TWO]
[4]   OUT←0 ¯1↓OUT◇TEXTSIZE 12◇TEXTFONT ZERO◇TEXTFACE ZERO◇MS←TEXTWIDTH MSG
[5]   SIZE←11 6×1 2+⍴OUT◇ULC←60 100◇C1←L1,L2,L3,L4,L5,L6◇P←⍳SIX
[6]   SIZE[TWO]←SIZE[TWO]⌊246◇LC←ZERO◇RC←⌊(¯9+SIZE[TWO])÷SIX◇S←IZ
[7]   B4:S1←ULC[ONE]◇S2←ULC[TWO]◇S3←S1+SIZE[ONE]◇S4←S2+SIZE[TWO]◇K←⎕EX 'UN2'
[8]   RECT←S1,S2,S3,S4◇B←¯18 0 18 0+RECT◇UN←TWO↑B◇UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
[9]   UN2←⎕GETBITS UN◇OPENPICTURE B◇ERASERECT B◇TR←¯18 0 1 0+S1,S2,S1,S4
[10]  CR←¯14 10 ¯3 22+S1,S2,S1,S2◇G3←¯1 0 18 18+S3,S2,S3,S2
[11]  G3 DRAWPICTURE LEFTARROW◇G4←¯1 ¯18 18 0+S3,S4,S3,S4
[12]  G4 DRAWPICTURE RIGHTARROW◇K←(S1-15)+TWO×⍳SIX
[13]  K←⍉4 6⍴(K-ONE),(SIX⍴TWO+S2),K,SIX⍴S4+NTWO◇RECT←RECT⍪5 4⍴CR,TR,G3,G4
[14]  FRAMERECT K⍪B⍪RECT◇G1←⌊PFIVE×S4+S2-MS⌊S4-S2+30
[15]  K←(S1-17),G1,(S1-ONE),S2+S4-G1
[16]  ERASERECT K⍪3 4⍴(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
[17]  MOVETO (S1-THREE),G1◇TEXTFONT ZERO◇TEXTSIZE 12◇MSG DRAWMSG MS,MS⌊S4-S2+30
[18]  TEXTNORMAL◇GRAYPAT FILLRECT 0 18 17 ¯18+S3,S2,S3,S4
[19]  B14:MOVETO 10 8+S1,S2◇DRAWTEXT (N,RC)↑(ZERO,LC)↓OUT
[20]  B18:G5←(RC×¯36+S4-S2)÷TWO×(⍴OUT)[TWO]
[21]  G2←S2+N+((LC+PFIVE×RC)÷(⍴OUT)[TWO])×S4-S2+36
[22]  G5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◇ERASERECT G5◇FRAMERECT G5
[23]  RECT[SIX;]←G5◇B DRAWPICTURE CLOSEPICTURE
[24]  B2:M←⎕GETKEY◇→(THREE≠⍴M)/B2◇→(TWO=M[ONE])/B2◇M←M[2 3]
```

```
[25]   L←ONE↑(M PTINRECT RECT)/P◊→(ZERO=L)/B2◊→C1[L]
[26]   B3:⎕SOUND BEEP◊→B2
[27]   L1:→BUTTON/L1◊G3←⌈((NTWO+M[ONE])-S1)÷11◊→(G3<ONE)/B2◊→(G3>N)/B2
[28]   G4←+/FL<LC+(M[TWO]-S2+FIVE)÷SIX◊K←G3+N×G4-1◊→(JL<K)/B2◊→(K∊A)/B1◊S←K
[29]   L2:→BUTTON/L2◊UN2 ⎕PUTBITS UN◊→ZERO
[30]   B1:K←'This Job Has Already Been Scheduled Or Is Being Scheduled'
[31]   ERRORWINDOW K◊→B2
[32]   L3:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
[33]   B7:→(~BUTTON)/B5◊FRAMERECT M1◊M1←B+FOURρ(0 0⌈GETMOUSE)-M◊FRAMERECT M1◊→B7
[34]   B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
[35]   B30:FRAMERECT M1◊UN2 ⎕PUTBITS UN◊PENPAT BLACKPAT◊PENMODE EIGHT◊→B4
[36]   L4:→(LC≤ZERO)/B3◊HS←(⌽(LC-ONE)↑OUT[ONE;])ιAV125◊G3←LC⌊HS
[37]   SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO◊LC←LC-G3
[38]   OPENPICTURE B◊MOVETO 10 8+S1,S2◊DRAWTEXT (N,G3⌊RC)↑(ZERO,LC)↓OUT
[39]   GRAYPAT FILLRECT G5◊→B18
[40]   L5:G3←(ρOUT)[TWO]-RC◊→(LC≥G3)/B3◊HS←((RC+LC+ONE)↓OUT[ONE;])ιAV125◊G3←HS⌊G3
[41]   G3←HS⌊G3⌊RC◊SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(¯6×G3),ZERO
[42]   LC←LC+G3◊OPENPICTURE B◊MOVETO (S1+TEN),S2+EIGHT+SIX×RC-G3
[43]   DRAWTEXT (N,-G3)↑(N,RC)↑(ZERO,LC)↓OUT◊GRAYPAT FILLRECT G5◊→B18
[44]   L6:M1←G5◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[45]   B23:→(~BUTTON)/B24◊FRAMERECT M1
[46]   K←(18+S2-G5[2])⌈(S4-18+G5[4])⌊1↓GETMOUSE-M◊M1←G5+4ρ0,K◊FRAMERECT M1◊→B23
[47]   B24:FRAMERECT M1◊OPENPICTURE B
[48]   GRAYPAT FILLRECT G5◊PENPAT BLACKPAT◊PENMODE EIGHT
[49]   LC←LC+⌊PFIVE+K×(ρOUT)[TWO]+S4-S2+36◊ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◊→B14
     ∇


     ∇Z←B CHARMAT V;A
[1]    V←V,(,B)[ONE]
[2]    Z←(ρA)ρ(,A←A∘.≥ι⌈/0,A←(A≠0)/A←A-1+0,NONE↓A←A/ιρA)\(~A←V∊B)/V
     ∇


     ∇S←G1 CHOICE2 A;M;N;M1;RC;E;FL;HS;SIZE;P;K;L
[1]    S←G1 CH2 A
     ∇


     ∇CLEAR;K;KK
[1]    K←ONE◊HIDECURSOR◊KK←503132+524288×⌊PFIVE+⎕WSSIZE÷524288
[2]    CLIPRECT 0 0 299 507
[3]    B1:MFIVEHEADER[K;] ⎕POKE KK+K×64◊K←K+ONE◊→(K≤NINE)/B1◊SHOWCURSOR
     ∇


     ∇CLEARBUFFER
[1]    B1:→(ZERO≠ρ⎕GETKEY)/B1
     ∇


   ∇CLIPRECT∇


     ∇A CLIST B;C;K
[1]    →(ZERO=×/ρA)/ZERO◊ERASERECT B◊TEXTFACE ONE◊A←(NTWO↑ONE,ρA)ρA
[2]    C←(EIGHT+⌈PFIVE×B[THREE]+B[ONE])-FIVE×ONE↑ρA◊K←ZERO
[3]    MOVETO C,B[TWO]+FIVE◊DRAWTEXT A◊TEXTFACE ZERO
     ∇
```

```
∇CLOSEPICTURE∇


     ∇CONPROC;K;G1;G2;KK;G3
[1]    K←(ρSP)[ONE],FOUR◊TL←+/(Kρ¯32 ¯64 ¯128 ¯256)×SP[;CL4]>KρRESX[JS;]
[2]    TL←TL+¯16×ZERO>SP[;CL5]×AUDX[JS]◊G1←−CONP[ONE+JS;ONE]◊G2←T3[JS]
[3]    SP1←SP[;CL6]◊G3←(NONE+TARι-JS+ONE)↑TAR◊G3←(G3ι-JS)↓G3
[4]    KK←G1,G2,G3
[5]    K←(~KKεSP1)/KK◊→(ZERO=ρK)/B1◊K←((ιρK)=KιK)/K◊KK←SP1,K
[6]    SP1←KK[⍋KK]◊K←+\~SP1εK◊SPA←SP[K;ICL]◊TL←TL[K]◊→B2
[7]    B1:SPA←SP[;ICL]
[8]    B2:TL←TL+(NTWO×SP1<G1)+NFOUR×SP1≥G2◊→(ZERO=ρG3)/ZERO
[9]    G3←((PFIVE×ρG3),TWO)ρG3◊TL←TL+NEIGHT×∨/(SP1∘.≥G3[;ONE])∧SP1∘.<G3[;TWO]
     ∇


     ∇TET←DEF CONVT3 DATE;VI;FI;DAY;MIN;HR;M;N;A
[1]    M←(DEF=ZERO)+×DEF◊TET←IZ◊N←0 24 60⊤|DEF◊DATE[(DATEε'/,:+')/ιρDATE]←' '
[2]    DATE[(DATE='-')/ιρDATE]←'¯'◊DATE←(~(∧\DATEε' ¯')∧DATE=' ')/DATE
[3]    VI←⎕VI DATE◊→(ZEROεVI)/ZERO◊→(ZERO=ρVI)/ZERO
[4]    →(THREE<ρVI)/ZERO◊FI←⎕FI DATE◊→(ZERO>NONE↑FI)/ZERO◊→(60≤NONE↑FI)/ZERO
[5]    →(∨/FI≠⌊FI)/ZERO◊A←ONE-TWO×'¯'=ONE↑(DATE≠' ')/DATE
[6]    →(ONE≠ρVI)/B1◊TET←M×FI+60×N[TWO]+24×N[ONE]◊→ZERO
[7]    B1:→(THREE=ρFI)/B3◊→(A=NONE)/B4
[8]    B5:→(24≤FI[ONE])/ZERO◊TET←M×FI[TWO]+60×FI[ONE]+24×N[ONE]◊→ZERO
[9]    B4:→((TET≠ZERO)∧N[ONE]≠ZERO)/ZERO◊M←NONE◊FI[ONE]←|FI[ONE]◊→B5
[10]   B3:→(ZERO>FI[TWO])/ZERO◊→(24≤FI[TWO])/ZERO
[11]   TET←A×FI[THREE]+60×FI[TWO]+24×|FI[ONE]
     ∇


     ∇CSCHED;SPB;G2;VAL;K;G1;G3;G5;KK;N;P;G4;A;G6
[1]    →(ZERO=CS)/B13◊→(STR[JS]≥ZERO)/B30◊G4←(MD+ONE),NONE◊K←ONE◊P←T2[JS]
[2]    SPB←ZERO=SPA-⍉(CL,ρTL)ρTL◊→(ONE=CS)/B2
[3]    G2←(PDS≥ZERO)∧~PR[JS;]◊G6←PR[JS;]/ICL◊G2←G2/ICL◊→(CS=ρG6)/B14
[4]    →(ZERO=ρG2)/B10◊G1←G2[⍎'C',(⍕CS-ρG6),⍕ρG2]◊N←⍋+/PDS[G1]+WL[G1]
[5]    B4:G3←G6,G1[N[K];]
[6]    B15:VAL←PDS[G3]◊G2←VAL ISEC SPB[;G3]◊→(G2>P)/B12
[7]    →(G4[ONE]≤G2+⌈/VAL)/B12◊G4←(G2+⌈/VAL),G2,G3
[8]    B12:K←K+ONE◊→(K≤ρN)/B4◊G2←G4[TWO]◊→(NONE=G2)/B10◊G3←TWO↓G4◊→B6
[9]    B14:G3←G6◊N←,ONE◊→B15
[10]   B2:G1←,PR[JS;]/ICL◊SPB←SPB≠ZERO,¯1 0↓SPB◊→(ZERO≠ρG1)/B7◊G1←⍋WL+PDS
[11]   B7:G3←,G1[K]◊VAL←PDS[G3]◊→(VAL<ZERO)/B8◊KK←SPB[;IZρG3]/SP1
[12]   N←⌊PFIVE×ρKK◊KK←(N,TWO)ρKK◊G5←((-VAL)≥-/KK)ιONE
[13]   →(G5>N)/B8◊G2←KK[G5;ONE]◊→(G2>P)/B8◊→(G4[ONE]≤G2+VAL)/B8◊G4←(G2+VAL),G2,G3
[14]   B8:K←K+ONE◊→(K≤ρG1)/B7◊G2←G4[TWO]◊→(NONE=G2)/B10◊G3←TWO↓G4
[15]   B6:→AP/B1◊VAL←PDS[G3]◊→(~JSεJSS)/B3◊OPENPICTURE SCREEN
[16]   G4←JS◊JS←IZ◊PRS 0 1◊JS←G4◊SCREEN DRAWPICTURE CLOSEPICTURE◊M4←0 4ρIZ
[17]   B3:TEXTFACE ZERO◊20 460 DRAWTIME G2
[18]   →(G2>SS+360×PIX)/B1◊G1←16+18×G3◊K←ONE◊G5←G1+SEVEN
[19]   G1←G1,((ρG3)ρ⌊15.5+(ZERO⌈G2-SS)÷PIX),G5,376⌊⌊15.5+(ZERO⌈VAL+G2-SS)÷PIX
[20]   G1←⍉(FOUR,ρG3)ρG1◊GRAYPAT FILLRECT G1◊FRAMERECT G1
[21]   B9:→(SS≥G2+VAL[K])/B11◊MOVETO ¯2 ¯2+TWO↑G1[K;]◊DRAWTEXT 'J',⍕JS
[22]   B11:K←K+ONE◊→(K≤ρG3)/B9
[23]   B1:G3 ASSIGN G2,PDS+G2◊→ZERO
[24]   B10:→AP/ZERO◊→(ACS∧AJS)/ZERO
[25]   ERRORWINDOW 'I Am Unable To Schedule This Job'◊→(JSεJSS)/ZERO
[26]   AJS←ZERO◊ACS←ZERO◊PLANMENU[2 4;TWO]←' '◊PLANN SETMENU PLANMENU◊→ZERO
[27]   B30:G2←STR[JS]◊KK←NTWO+(SP1>G2)ιONE◊G4←KK↓SP1◊G6←(PDS≥ZERO)/ICL
```

```
 G5←+/(G2+PDS[G6])∘.>G4⋄G1←KK+ι⌈/G5⋄K←ZERO=(+\TL[G1])[G5]⋄G6←K/G6
 →(CS>ρG6)/B10⋄G6←(ZERO=1 1⍉(+\SPA[G1;G6])[K/G5;ιρG6])/G6⋄→(CS>ρG6)/B10
 G1←PR[JS;]/ICL⋄→(~∧/G1∊G6)/B10
 B34:G3←G1⋄→(CS=ρG3)/B6⋄→(CS>ONE)/B31⋄G6←G6[⍋WL[G6]+PDS[G6]]
 G3←,G6[PDS[G6]ι⌊/PDS[G6]]⋄→B6
 B31:G6←(~G6∊G1)/G6⋄G3←G6[⍎'C',(⍕CS-ρG1),⍕ρG6]
 G3←G3[⍋+/PDS[G3]+WL[G3];]⋄G4←⌈/PDS[G3]⋄G3←G1,G3[G4ι⌊/G4;]⋄→B6
 B13:SPB←ZERO=TL⋄→(STR[JS]≥ZERO)/B16⋄SPB←SPB≠ZERO,NONE↓SPB
 KK←SPB/SP1⋄N←⌊PFIVE×ρKK⋄KK←(N,TWO)ρKK⋄G5←((-ONE↑PDS)≥-/KK)ιONE⋄→(G5>N)/B10
 G3←,EIGHT⋄G2←KK[G5;ONE]⋄→B6
 B16:G2←STR[JS]⋄G4←SP1>G2⋄→(ZERO≠TL[NONE+G4ιONE])/B10
 →(ZERO≠+/(G4∧SP1<G2+ONE↑PDS)/TL)/B10⋄G3←,EIGHT⋄→B6
∇


∇CUR A
 ⎕CURSOR←(⎕CURSOR)[1],A
∇


⍒CUTPICTURE⍒


∇R←CVTTD T;G;G1
 T←,T⋄G1←((ρT),ONE)ρ' -'[ONE+T<ZERO]⋄R←⍉0 24 60⊤|T⋄G←(ρR)[ONE],NTWO
 R←G1,(⍕R[;,ONE]),'/',(G↑⍕100+R[;,TWO]),':',G↑⍕100+R[;,THREE]
∇


∇R←DEF CVTTIME1 T;DAY;YR;RD;F;M
 →(DEF=T)/B1⋄T←,0 365.25 24 60⊤T⋄YR←',',⍕1900+T[ONE]
 F←(12-24×ZERO=FOUR|T[ONE])↑,MOS
 M←ONE++/(12ρONE+⌊T[TWO])>F⋄DAY←' ',NTWO↑⍕ONE+⌊T[TWO]-(ZERO,F)[M]
 R←MONTHS[M;],DAY,YR,' ',(NTWO↑⍕T[THREE]),':',NTWO↑⍕T[FOUR]+100⋄→ZERO
 B1:R←'**NO INFO**'
∇


∇DATALOAD W0;W1;W2;W3;W4;W5;W7
 W1←ONE+⌈/ZERO,⎕FNUMS⋄W0 ⎕FHTIE W1
 B1:W3←NONE↓⎕FREAD W1⋄→(ZERO=ρW3)/B3⋄W4←NONE↓⎕FREAD W1⋄W5←ONE↑⎕FREAD W1
 W2←NONE↓⎕FREAD W1⋄→(W5='N')/B2⋄→(W5='F')/B4
 B5:W2←(⎕FI W4)ρW2⋄⍎W3,'←W2'⋄→B1
 B2:W2←(⎕FI W4)ρ⎕FI W2⋄⍎W3,'←W2'⋄→B1
 B4:W2←(⎕FI W4)ρW2⋄W3←⎕FX W2⋄→B1
 B3:⎕FUNTIE W1
∇


∇K←W7 DATASAVE KK;G1;G2;G3;G4;G5;G6;G7;⎕ELX
 ⎕ELX←'⎕ERROR((⎕DMι⎕TCNL)-⎕IO)↑⎕DM'
 G4←(ρKK)[ONE]⋄G1←ONE+⌈/ZERO,⎕FNUMS⋄G2←ONE-(⌽W7)ι':'⋄K←G2↓W7⋄G3←⎕LIB K
 K←((ρG3)[ONE],ρK)ρUPPERCASE K),G3
 G1←⌈/(ρW7),ONE↓ρK⋄→((((ρK)[ONE],G1)↑K) FIND (ONE,G1)ρG1↑UPPERCASE W7)/B5
 B7:W7 ⎕FHCREATE G1⋄K←ONE⋄G5←⎕TCNL⋄G6←G5,'C',G5⋄G7←G5,'N',G5
 B1:G2←DLBS KK[K;]⋄→(3=⎕NC G2)/B4⋄G3←⍎G2⋄→(82≠⎕DR G3)/B2
 (G2,G5,(⍕ρG3),G6,(,G3),G5) ⎕FAPPEND G1⋄K←K+ONE⋄→(K≤G4)/B1⋄→B8
 B2:(G2,G5,(⍕ρG3),G7,(⍕,G3),G5) ⎕FAPPEND G1⋄K←K+ONE⋄→(K≤G4)/B1⋄→B8
 B4:G3←⎕CR G2⋄(G2,G5,(⍕ρG3),G5,'F',G5,(,G3),G5) ⎕FAPPEND G1⋄K←K+ONE
```

```
[10]  →(K≤G4)/B1
[11]  B8:⎕FUNTIE G1◊→ZERO
[12]  B5:W7 ⎕FHTIE G1◊W7 ⎕FERASE G1◊→B7
     ∇
```

∇DELETEMENU∇

```
     ∇DELETEMENUS
[1]   DELETEMENU OPTIONSN,PLANN,CONSTN,MOUSEN◊DRAWMENUBAR
     ∇
```

```
     ∇R←DLBS A
[1]   R←(NONE+A⍳' ')↑A
     ∇
```

```
     ∇R←DLBS2 A
[1]   R←(-+/∧\' '=⌽A)↓A
     ∇
```

```
     ∇R←DLBS3 A
[1]   R←(ZERO,-+/∧\⌽∧≠' '=A)↓A
     ∇
```

∇DRAWLINE∇

∇DRAWMENUBAR∇

```
     ∇A DRAWMSG B
[1]   →(B[ONE]=B[TWO])/B1◊B←B[TWO]
[2]   B2:A←NONE↓A◊→(B<TEXTWIDTH A)/B2
[3]   B1:DRAWTEXT A
     ∇
```

∇DRAWPICTURE∇

∇DRAWTEXT∇

```
     ∇A DRAWTIME B
[1]   MOVETO A◊B←0 24 60⊤IZ⍴B◊DRAWTEXT (NTWO↑⍕B[TWO]),':',NTWO↑⍕100+NONE↑B
     ∇
```

```
     ∇Q←ED2A;SROW;SCOL;C1;C2;G1;K;G2;G3;G4;H;S3
[1]   H←(⍴LIST)[TWO]◊Q←ONE◊SROW←IZ⍴⌈((NTWO+M[ONE])-S1)÷ELEVEN
[2]   →(SROW<ONE)/ZERO◊→(SROW>N)/ZERO◊SCOL←+/FL<LC+(M[TWO]-S2+FIVE)÷SIX
[3]   C1←IZ⍴SROW+N×SCOL-ONE◊S3←⍎RPROG◊→(ZERO=⍴S3)/B10
```

```
→(AMAX<C1)/ZERO◊K←,LIST[C1;]◊G2←(ONE-(⌽K)ι'/')↓K
G3←S2+SEVEN+SIX×ZERO⌈FL[SCOL]-LC◊C2←ONE+S2+SIX×(RC+ONE)⌊FL[SCOL+ONE]-LC
G4←(¯10+S1+SROW×ELEVEN),G3,(ONE+S1+SROW×ELEVEN),C2◊INVERTRECT G4
B3:G1←4 1 0 0 REPLYWINDOW S3◊→(OPT=2 3 6)/B6,B2,B5◊S[C1]←G1
G1←(⍕G1),' '◊G1←G2,(-(ρG1)⌈H-ρG2)↑G1◊→((ρG1)>ONE+H)/B1
B8:OUT[SROW;FL[SCOL]+ιH]←LIST[C1;]←H↑G1
K←S1+NONE+ELEVEN×SROW◊ERASERECT ¯9 3 2 ¯1+K,S2,K,S4
MOVETO K,EIGHT+S2◊DRAWTEXT RC↑LC↓,OUT[SROW;]◊→ZERO
B1:LIST←DLBS3 LIST[ιC1-ONE;] APPEND G1 APPEND (C1,ZERO)↓LIST
UN2 ⎕PUTBITS UN◊Q←ZERO◊→ZERO
B6:INVERTRECT G4◊→ZERO
B2:G1←((NONE+H)↑G2),'*'◊S[C1]←NONE◊→B8
B5:HELPWINDOW HELPMSG◊→B3
B10:ERRORWINDOW MSG3◊→ZERO
∇


∇S←C EDA R;K;G1;G2;G3;G4;G5;L;M;N;M1;RC;E;FL;LIST;MSG;S;SIZE;HS;RPROG;P2;C1
 S←C EDITA R
∇


∇S←P EDITA Q;AMAX;S1;S2;S3;S4;OUT;LC;A;ULC;B;RECT;MS;CR;TR;C;OUT2;UN;UN2;ML
 N←Qι' '◊MSG←N↓Q◊LIST←⍎Q←N↑Q◊AMAX←(ρLIST)[ONE]◊N←18◊ULC←60 100
 K←MSGι' '◊RPROG←K↑MSG◊MSG←' ',(K↓MSG),' '◊S←P
 TEXTSIZE 12◊TEXTFONT ZERO◊TEXTFACE ZERO◊MS←TEXTWIDTH MSG◊ML←TEXTWIDTH MSG2
 P2←ιSEVEN◊C1←L1,L2,L13,L3,L4,L5,L6◊LC←ZERO
 B40:OUT←18 0ρIZ◊K←ZERO
 B31:OUT←OUT ADJOIN2 LIST[K+ιN⌊AMAX-K;],AV125◊K←K+N◊→(AMAX>K)/B31
 OUT←0 ¯1↓OUT◊→(20<(ρOUT)[TWO])/B45◊OUT←18 20↑OUT
 B45:OUT2←OUT◊FL←ZERO,((OUT[ONE;]=AV125)/ι(ρOUT)[TWO]),ONE+(ρOUT)[TWO]
 SIZE←11 6×1 2+ρOUT◊SIZE[TWO]←(ML+TEN)⌈(MS+50)⌈SIZE[2]⌊360
 RC←⌊(¯9+SIZE[TWO])÷SIX
 B4:S1←ULC[ONE]◊S2←ULC[TWO]◊S3←S1+SIZE[ONE]◊S4←S2+SIZE[TWO]
 RECT←S1,S2,S3,S4◊B←¯36 0 18 0+RECT◊UN←TWO↑B◊UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
 K←⎕EX 'UN2'◊UN2←⎕GETBITS UN◊ERASERECT B◊TR←¯36 0 ¯17 0+S1,S2,S1,S4
 CR←¯32 10 ¯21 22+S1,S2,S1,S2◊G3←¯1 0 18 18+S3,S2,S3,S2
 G3 DRAWPICTURE LEFTARROW◊G4←¯1 ¯18 18 0+S3,S4,S3,S4
 G4 DRAWPICTURE RIGHTARROW◊K←(S1-33)+TWO×ιSIX◊C←¯32 ¯16 ¯21 ¯4+S1,S4,S1,S4
 K←⍉4 6ρ(K-ONE),(SIXρTWO+S2),K,SIXρS4+NTWO◊RECT←RECT⍪6 4ρCR,C,TR,G3,G4
 FRAMERECT K⍪B⍪RECT◊G1←⌊PFIVE×S4+S2-MS⌊S4-S2+45◊TEXTSIZE 12
 K←(S1-35),G1,(S1-19),(S4+S2-G1),0 ¯1 1 0 0 0 1 1+C[1 2 3 2 1 4 3 4]
 ERASERECT 6 4ρK,(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
 MOVETO (S1-21),G1◊TEXTFONT ZERO◊MSG DRAWMSG MS,MS⌊S4-S2+45
 MOVETO (S1-THREE),⌊PFIVE×S4+S2-ML⌊S4-S2◊MSG2 DRAWMSG ML,ML⌊S4-S2
 TEXTNORMAL◊GRAYPAT FILLRECT 2 4ρ(C+1 1 ¯1 ¯1),0 18 17 ¯18+S3,S2,S3,S4
 B14:MOVETO 10 8+S1,S2◊DRAWTEXT (N,RC)↑(ZERO,LC)↓OUT
 B18:G5←(RC×S4-S2+36)÷TWO×(ρOUT)[TWO]
 G2←S2+N+((LC+PFIVE×RC)÷(ρOUT)[TWO])×S4-S2+36
 G5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◊ERASERECT G5◊FRAMERECT G5
 RECT[SEVEN;]←G5
 B2:M←⎕GETKEY◊→(THREE≠ρM)/B2◊→(TWO=M[ONE])/B2◊M←M[2 3]
 L←ONE↑(M PTINRECT RECT)/P2◊→(ZERO=L)/B2◊→C1[L]
 B3:⎕SOUND BEEP◊→B2
 L1:→BUTTON/L1◊K←ED2A◊→K/B2◊→B40
 L2:→BUTTON/L2◊→(~GETMOUSE PTINRECT RECT[L;])/B2◊UN2 ⎕PUTBITS UN◊→ZERO
 L3:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
 B7:→(~BUTTON)/B5◊FRAMERECT M1◊M1←B+FOURρ(0 0⌈GETMOUSE)-M◊FRAMERECT M1◊→B7
 B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
 B30:FRAMERECT M1◊UN2 ⎕PUTBITS UN◊PENPAT BLACKPAT◊PENMODE EIGHT◊→B4
 L4:→(LC≤ZERO)/B3◊HS←(⌽(LC-ONE)↑OUT[ONE;])ιAV125◊G3←LC⌊HS
 SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO◊LC←LC-G3
```

```
MOVETO 10 8+S1,S2◇DRAWTEXT (N,G3⌊RC)↑(ZERO,LC)↓OUT◇GRAYPAT FILLRECT G5
→B18
L5:G3←(ρOUT)[TWO]-RC◇→(LC≥G3)/B3◇HS←((RC+LC+ONE)↓OUT[ONE;])⍳AV125◇G3←HS⌊G3
SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(¯6×G3),ZERO◇LC←LC+G3
MOVETO (S1+TEN),S2+EIGHT+SIX×RC-G3◇DRAWTEXT (N,-G3)↑(N,RC)↑(ZERO,LC)↓OUT
GRAYPAT FILLRECT G5◇→B18
L6:M1←G5◇PENPAT GRAYPAT◇PENMODE TEN◇K←ZERO
B23:→(~BUTTON)/B24◇FRAMERECT M1
K←(18+S2-G5[2])⌈(S4-18+G5[4])⌊ONE↓GETMOUSE-M◇M1←G5+4ρ0,K◇FRAMERECT M1◇→B23
B24:FRAMERECT M1◇GRAYPAT FILLRECT G5◇PENPAT BLACKPAT◇PENMODE EIGHT
LC←LC+⌊PFIVE+K×(ρOUT)[TWO]÷S4-S2+36◇ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◇→B14
L13:S←P◇UN2 ⎕PUTBITS UN◇→ZERO
∇


∇EF A;R;G1;R;SIZE;T;J;V;R1
J←ONE+⌈/ZERO,⎕FNUMS◇A ⎕FHTIE J◇A←(ONE-(⌽A)⍳':')↑A◇V←A,'VARS'
B1:G1←NONE↓⎕FREAD J◇→(ZERO=ρG1)/B3◇SIZE←NONE↓⎕FREAD J◇T←ONE↑⎕FREAD J
R←NONE↓⎕FREAD J◇V←V APPEND A,G1◇→(T='N')/B2
B5:R←(⎕FI SIZE)ρR◇⍎A,G1,'←R'◇→B1
B2:R←(⎕FI SIZE)ρ⎕FI R◇⍎A,G1,'←R'◇→B1
B3:(WS,SCHEDDISK,A) ⎕FERASE J◇⍎A,'VARS←V'◇→ZERO
∇


∇R←A ELIM B
R←(~(⍳(ρA)[ONE])∊,B)⌿A
∇


∇ERASERECT∇


∇ERRORWINDOW A;OS;K;M;D;UN;UN2;KK;P
D←22 20 70 480◇UN←22 20 70 484◇KK←ONE
UN2←⎕GETBITS UN◇ERASERECT D◇FRAMERECT 2 4ρD,24 22 68 478
FRAMEROUNDRECT 2 6ρ33 385 57 460 15 15 31 383 59 462 17 17
TEXTSIZE 18◇TEXTFACE 65◇MOVETO 52 407◇DRAWTEXT 'OK'◇⎕SOUND BEEP
TEXTSIZE NINE◇TEXTFACE ONE◇K←35+FIVE×FOUR-THREE⌊⌈(ρA)÷45
B1:MOVETO K,50◇OS←45-(⌽45↑A)⍳' '◇P←OS<ZERO◇OS←OS+45×P◇→((KK=3)∧OS<ρA)/B5
DRAWTEXT OS↑A◇KK←KK+ONE◇A←(OS+~P)↓A◇K←K+ELEVEN◇→(ZERO≠ρA)/B1◇TEXTFACE ZERO
B2:SHOWCURSOR◇M←⎕GETKEY◇→(THREE≠ρM)/B2◇→(TWO=M[ONE])/B2
→(~M[2 3] PTINRECT 33 385 57 460)/B3◇UN2 ⎕PUTBITS UN
B4:→(ZERO≠ρ⎕GETKEY)/B4◇→ZERO
B3:SHOWCURSOR◇→BUTTON/B3◇→B2
B5:DRAWTEXT (41↑A),'...'◇TEXTFACE ZERO◇→B2
∇


∇R←FILEOPEN P
B1:R←⎕SFOPEN 'TEXT'◇CLEAR◇→(ZERO=ρR)/ZERO◇→(P≡(ρP)↑(ONE-(⌽R)⍳':')↑R)/ZERO
ERRORWINDOW 'Please Select A File Whose Name Begins With :  ',(¯1↓P)◇→B1
∇


∇R←A FILESAVE B;G1
→(~':'∊B)/B2◇B←(B⍳':')↓B
B2:G1←B⍳'_'
B1:R←A ⎕SFSAVE B◇CLEAR◇→(ZERO=ρR)/ZERO◇→((G1↑B)≡G1↑(ONE-(⌽R)⍳':')↑R)/ZERO
```

```
[4]   ERRORWINDOW 'Please Begin File Name With :  ',(¯1↓B),'_'◊→B1
     ∇
```

∇FILLRECT∇

```
     ∇R←A FIND B;D
[1]   D←⌈/(ρA)[TWO],(ρB)[TWO]◊R←∨≠(((ρA)[ONE],D)↑A)∧.=⍉((ρB)[ONE],D)↑B
     ∇
```

```
     ∇R←W FIND3 T
[1]   R←(∨/∧≠(⍉((ρT)[ONE],ρW)ρNONE+ιρW)⌽W∘.=T)/ι(ρT)[ONE]
     ∇
```

∇FRAMERECT∇

∇FRAMEROUNDRECT∇

```
     ∇R←B GETINPUT A;ST;G1;G2;G3;LASTP;CURP;E;A1;A2;D;KK;UR;C1;C2;AA
[1]   AA←TWO⌈NONE+⌊(A[FOUR]-A[TWO])+SIX
[2]   L1:OPT←ONE◊ST←(THREE+⌈PFIVE×A[THREE]+A[ONE]),FIVE+A[TWO]
[3]   ERASERECT A+0 0 3 0◊B←,B◊HIDECURSOR◊G1←ONE↑B◊→(6 7=G1)/0 0
[4]   G2←~(ONE=G1)×(TWO↑B)[TWO]◊G3←(3↑B)[3]×ONE=G1◊C1←¯8 0 2 0◊C2←13
[5]   R←''◊LASTP←ST◊TEXTFACE ONE◊MOVETO ST◊DRAWTEXT AV125
[6]   CURP←GETPEN◊A1←210ρ'→(ONE=ρE←⎕GETKEY)/B4◊'◊A2←210ρ'→(ONE=ρE←⎕GETKEY)/B5◊'
[7]   B1:E←⎕GETKEY◊→(ONE=ρE)/B4◊⍎A1◊ERASERECT C1+ST,CURP
[8]   E←⎕GETKEY◊→(ONE=ρE)/B5◊⍎A2◊MOVETO ST◊DRAWTEXT AV125◊→B1
[9]   B4:ERASERECT C1+ST,CURP
[10]  B5:MOVETO ST◊→(C2=E)/B3◊→(EIGHT=E)/B2◊→(E>255)/B1
[11]  KK←AV[E+ONE]◊→G2/B6◊→(~KKεVAL)/B16
[12]  B6:D←KK◊R←R,D◊TEXTFACE ZERO◊DRAWTEXT D◊→(AA≤ρR)/B7◊ST←GETPEN◊LASTP←LASTP,ST
[13]  B9:→(G3=ρR)/B3◊TEXTFACE ONE◊DRAWTEXT AV125◊CURP←GETPEN◊→B1
[14]  B7:SCROLLRECT (¯1 5 2 0+A),¯6 0◊→(G3=ρR)/B3◊TEXTFACE ONE◊→B1
[15]  B10:SCROLLRECT (¯1 5 2 0+A),6 0◊MOVETO TWO↑LASTP
[16]  TEXTFACE ZERO◊DRAWTEXT R[TWO+(ρR)-AA]◊TEXTFACE ONE◊→B14
[17]  B16:⎕SOUND BEEP◊→B1
[18]  B2:→(ZERO=ρR)/B16◊R←NONE↓R◊ERASERECT C1+(TWO↑NFOUR↑ST,LASTP),ST
[19]  →(AA≤ONE+ρR)/B10◊LASTP←(TWO⌈NTWO+ρLASTP)↑LASTP
[20]  B14:MOVETO ST←NTWO↑LASTP◊DRAWTEXT AV125◊CURP←GETPEN◊→B1
[21]  B3:UR←UPPERCASE R◊OPT←ONE+(ZERO=ρR)+(FIVE×UR≡'HELP')+TWO×UR≡'DELETE'
[22]  TEXTFACE ZERO◊INITCURSOR◊→(OPT≠ONE)/0◊→(1 2 3 4 5 8=G1)/0 0 0,L16,0,L2
[23]  L2:B←B,(ONE=ρB)/ZERO◊R←B[TWO] CONVT3 R◊→(ZERO≠ρ,R)/B21
[24]  ERRORWINDOW 'INVALID TIME'◊→L1
[25]  B21:(CVTTD R) CLIST A◊→(THREE>ρB)/ZERO◊→(B[THREE]≤R)/B20
[26]  ERRORWINDOW 'INVALID TIME.  EARLIEST VALID TIME IS ',,CVTTD B[THREE]◊→L1
[27]  B20:→(FOUR>ρB)/ZERO◊→(B[FOUR]≥R)/ZERO
[28]  ERRORWINDOW 'INVALID TIME.  LATEST VALID TIME IS ',,CVTTD B[FOUR]◊→L1
[29]  L16:→(~ZEROε,⎕VI R)ρL17
[30]  ERRORWINDOW 'INVALID ENTRY.  ENTER NUMERICAL VALUES ONLY.'◊→L1
[31]  L17:R←,⎕FI R◊(⍕R) CLIST A◊→(ONE=ρB)/0◊→(ZERO=B[TWO])/L18◊→(B[TWO]≥ρR)/L18
[32]  E←'TOO MANY ENTRIES.  ENTER AT MOST ',(⍕B[TWO]),(-ONE=B[TWO])↓' NUMBERS'
[33]  ERRORWINDOW E◊→L1
[34]  L18:→(TWO=ρB)/ZERO◊→(ZERO=B[THREE])/L19◊→(~ZEROεR=⌊R)/L19
[35]  ERRORWINDOW 'INVALID ENTRY.  ENTER INTEGERS ONLY.'◊→L1
[36]  L19:→(THREE=ρB)/ZERO◊→(FOUR=ρB)/L20◊→(B[FOUR]>B[FIVE])/L21
[37]  L20:→(~ONEεR<B[FOUR])/L21
```

```
[38]  ERRORWINDOW 'INVALID ENTRY.  ENTER VALUE GREATER THAN OR EQUAL TO ',⍕B[4]
[39]  →L1
[40]  L21:→(FOUR=ρB)/ZERO◊→(~ONE∊R>B[FIVE])/ZERO
[41]  ERRORWINDOW 'INVALID ENTRY.  ENTER VALUE LESS THAN OR EQUAL TO ',⍕B[5]◊→L1
    ∇
```

⍒GETMOUSE⍒

⍒GETPEN⍒

```
    ∇T GRAPH2 H;J;P;KK;F;E;G1;G3;G5;G4;G2;G6;N;V;K;V1
[1]   J←T[ONE]◊P←T[TWO]◊KK←T[THREE]◊T←THREE↓T◊F←H[;TWO]◊F←((F≥P⌊MD)⍳ONE)↑F
[2]   G1←NTWO+(F>J)⍳ONE◊E←G1↓⍳ρF◊F←G1↓F◊G3←ρF◊G3←TWO,G3
[3]   F[ONE]←J◊G4←T[TWO]◊V1←(T[4]-G4+TWO)÷P-J◊G2←G4+⌈(F-J)×V1
[4]   G2[G3]←G2[G3]⌊T[FOUR]-ONE◊N←ONE↓,⍉G3ρG2
[5]   G5←T[THREE]-ONE◊V←H[E;ONE]◊G6←G5-⌈V×(G5-T[ONE]+15)÷KK◊G1←NONE↓,⍉G3ρG6
[6]   FRAMERECT T◊PENPAT GRAYPAT◊DRAWLINE T[1 2 1 4]+15 1 15 ¯2
[7]   PENPAT BLACKPAT◊TEXTFACE ONE◊KK VCEN (15+T[ONE]),G4
[8]   P HCEN (G5+11),G4+⌈(P-J)×V1
[9]   MOVETO G5,G4◊LINETO ⍉G1,[PFIVE]N◊K←ONE◊G1←NONE
[10]  B1:J←G6[K]◊→(G1=J)/B2◊V[K] VCEN J,G4◊G1←J◊F[K] HCEN (G5+11),G2[K]
[11]  B2:K←K+ONE◊→(K<G3)/B1◊TEXTFACE ZERO
    ∇
```

```
    ∇A HCEN B
[1]   A←⍕A◊MOVETO B[ONE],B[TWO]+TWO-⌈3.5×ρA◊DRAWTEXT A
    ∇
```

```
    ∇HCL INS;K;J;FNS;VAR;TEXT;LINE;IDLIST;R;G;G1;M
[1]   →(INS≡'ALL')/L8◊VAR←0 0ρ''◊FNS←' ' CHARMAT ,INS◊→L9
[2]   L8:⎕EX 'INS'◊VAR←⎕NL TWO◊FNS←⎕NL THREE
[3]   L9:'ALIGN PRINTER PAPER; THEN PRESS RETURN'◊⎕SOUND BEEP
[4]   L7:→(ONE≠ρ⎕GETKEY)/L7◊⎕PRSELECT◊K←ZERO◊6ρ⎕TCNL
[5]   R←'Z←IDLIST L;W;B',⎕TCNL,'W←80-(ONE↓ρL←'' '',L)|80'
[6]   R←R,' ◊ B←(1↑(0,W)⊤×/ρL)+×|×/ρL ◊ Z←(B,W)ρ(B×W)↑,L ◊ Z←,Z,⎕TCNL '
[7]   0 0ρ⎕FX ⎕TCNL CHARMAT R
[8]   '     )FNS',⎕TCNL,(IDLIST FNS),2ρ⎕TCNL
[9]   L1:K←K+ONE◊→((ρFNS)[ONE]<K)/L6◊TEXT←⎕VR FNS[K;]
[10]  →(ZERO=ρTEXT)/L4◊TEXT,3ρ⎕TCNL◊→L1
[11]  L4:'     ⍒',(DLBS FNS[K;]),'⍒'◊''◊''◊→L1
[12]  L6:→(ZERO∊ρVAR)/L10◊'      )VARS',⎕TCNL,(IDLIST VAR),2ρ⎕TCNL
[13]  L10:⍞←⎕TCFF◊⎕PRUNSELECT◊⍞←'**DONE**'◊⎕SOUND BEEP
    ∇
```

```
    ∇HELPWINDOW A;OS;K;M;D;UN;UN2;KK;P;H;G1
[1]   A←⎕TCNL CHARMAT A
[2]   H←(ρA)[ONE]⌈THREE◊G1←22+16×⌈(18+11×H)÷16◊D←22 20,G1,480◊UN←22 20,G1,484
[3]   KK←ONE◊UN2←⎕GETBITS UN◊ERASERECT D◊FRAMERECT 2 4ρD,24 22,(G1-TWO),478
[4]   FRAMEROUNDRECT 2 6ρ33 385 57 460 15 15 31 383 59 462 17 17
[5]   TEXTSIZE 18◊TEXTFACE 65◊MOVETO 52 407◊DRAWTEXT 'OK'◊⎕SOUND BEEP
[6]   TEXTNORMAL◊TEXTFACE ZERO◊K←40
[7]   B1:MOVETO K,50◊DRAWTEXT A
[8]   B2:SHOWCURSOR◊M←⎕GETKEY◊→(THREE≠ρM)/B2◊→(TWO=M[ONE])/B2
[9]   →(~M[2 3] PTINRECT 33 385 57 460)/B3◊UN2 ⎕PUTBITS UN
```

```
[10]  B4:→(ZERO≠ρ⎕GETKEY)/B4◊→ZERO
[11]  B3:SHOWCURSOR◊→BUTTON/B3◊→B
    ∇


    ∇HIDECURSOR∇


    ∇INIT R;K;G1;G2;G3;G4;G5;RECT;SS;PIX;JON;CRN;NA;MS;ME;MD;RES;SP;M;L;A
[1]   B47:SETUP◊⎕ELX←'⎕DM'◊→(ZERO=ρR)/A7
[2]   B1:K←¯14+30×ι13◊KK←12 15 13 376⍪⍉(4 13)ρ(13ρ13),(K-ONE),(16+13ρ2 0),K
[3]   CLIPRECT 0 0 299 507◊ERASERECT 0 0 299 507◊TEXTMODE ONE
[4]   OPENPICTURE SCREEN◊TEXTFACE ONE◊MOVETO 198 365
[5]   CLEARBUFFER◊A←(TENρL11),L1,L2,L3,L4,L5,L6,L6,L6,L6,(8ρL10),(7ρL12),L13
[6]   DRAWTEXT 'FLT PLAN: ',NA◊MOVETO 207 365◊DRAWTEXT 'JOB:'◊MOVETO 176 2
[7]   DRAWTEXT MH◊K←ONE◊M1←41◊G1←'C',0 1↓⍕(CL,ONE)ρICL◊MOVETO 225 363
[8]   DRAWTEXT G1,' ',CRN◊MOVETO 176 460◊DRAWTEXT EIGHT↑VCEN2 +/PRI[JSS]
[9]   →(ZERO=ρJS)/B4◊MOVETO 33 443◊DRAWTEXT 'CREW ='◊MOVETO 32 490◊DRAWTEXT ⍕CS
[10]  MOVETO 216 370◊DRAWTEXT (⍕JS),' ',JON[JS;]◊FRAMERECT SR2
[11]  B4:→(ZERO=CL)/B3
[12]  B2:MOVETO M1,ZERO◊DRAWTEXT G1[K;]◊→(ZERO=ρJS)/B3◊→(ZERO=CS)/B3
[13]  →(ZERO>PDS[K])/B3◊MOVETO M1,378◊DRAWTEXT ⍕PDS[K]◊MOVETO M1,480
[14]  DRAWTEXT ⍕WL[K]
[15]  B3:K←K+ONE◊M1←M1+18◊→(CL≥K)/B2◊MOVETO 167 0◊DRAWTEXT 'C0'◊→(ZERO=ρJS)/B39
[16]  →(CS≠ZERO)/B39◊MOVETO 167 378◊DRAWTEXT ⍕ONE↑PDS◊MOVETO 167 480
[17]  K←(JT[JSS]=ZERO)/JSS◊DRAWTEXT ⍕+/ST1[K]-ST[K]
[18]  B39:¯1 420 17 439 DRAWPICTURE UPARROW
[19]  15 420 33 439 DRAWPICTURE DOWNARROW◊7 404 26 422 DRAWPICTURE LEFTARROW
[20]  7 437 26 455 DRAWPICTURE RIGHTARROW◊14 388 31 406 DRAWPICTURE CORNER
[21]  AA←A1,B1,A6,A7,A8,A9,A10◊AB←A2,A3,A4,A5,B53,B47,A15,A16,B47
[22]  FRAMERECT KK⍪9 456 24 498⍪SR⍪RECT[TEN+ιNINE;]
[23]  MOVETO 200,27◊DRAWTEXT 'POWER'◊MOVETO 200 93◊DRAWTEXT 'TRANSMISSION'
[24]  MOVETO 200 203◊DRAWTEXT 'MEMORY'◊MOVETO 200 285◊DRAWTEXT 'MULTIPLEX'
[25]  M1←3 4ρ179 15 186 376 179 415 186 430 179 471 186 486
[26]  MOVETO 186 0◊DRAWTEXT 'AC'◊MOVETO 186 385◊DRAWTEXT 'GEN     SEN'
[27]  STRIPEPAT FILLRECT M1[TWO;]◊GRAYPAT FILLRECT M1[THREE;]◊FRAMERECT M1
[28]  GRAYPAT FILLRECT 203 0 204 358
[29]  B9:TEXTFACE ZERO◊MOVETO 10 0◊DRAWTEXT ,8 10↑0 ¯3↓CVTTD SS+(60×PIX)×0,ιSIX
[30]  PRS ZERO◊SCREEN DRAWPICTURE CLOSEPICTURE◊TEXTMODE ZERO
[31]  B10:→TW/B11◊SW←CUTPICTURE B◊B DRAWPICTURE JW
[32]  B11:M←⎕GETKEY◊→(THREE≠ρM)/B11◊→(TWO=M[ONE])/A0◊M←M[2 3]
[33]  L←ONE↑(M PTINRECT RECT)/IR◊→(ZERO=L)/B13◊→TW/B12◊→((L≠2)∧L<11)/B12
[34]  →((L>15)∧L<20)/B12◊B DRAWPICTURE SW
[35]  B12:→A[L]
[36]  B13:L←ONE↑(M PTINRECT SRP[ICL,EIGHT;])/ICL,EIGHT◊→(ZERO=L)/B11◊→TW/B14
[37]  B DRAWPICTURE SW
[38]  B14:STATWINDOW IZρL◊→B10
[39]  B15:⎕SOUND BEEP◊→B10
[40]  L1:→(MD≤SS+PIX×360)/B15◊PIX←PIX+ONE◊OPENPICTURE SCREEN◊→(ZERO≠ρ,JS)/B17
[41]  B16:ERASERECT 0 0 11 401◊TEXTMODE ONE◊→B9
[42]  L2:→(PIX=ONE)/B15◊PIX←PIX-ONE◊OPENPICTURE SCREEN◊→(ZERO=ρJS)/B16
[43]  B17:ERASERECT SR2
[44]  RECT[19+ιCL⌈EIGHT×CS=ZERO;FOUR]←SR2[;FOUR]←400+ZERO⌈⌈,PDS+PIX
[45]  TEXTFACE ONE◊→(ZERO=CS)/B49◊K←ONE◊G2←23+18×ICL
[46]  B48:MOVETO G2[K],480◊DRAWTEXT ⍕WL[K]◊K←K+ONE◊→(CL≥K)/B48
[47]  FRAMERECT SR2◊→B16
[48]  B49:MOVETO 167 480◊K←(JT[JSS]=ZERO)/JSS◊DRAWTEXT ⍕+/ST1[K]-ST[K]
[49]  FRAMERECT SR2◊→B16
[50]  L3:→(SS=ZERO)/B15◊SS←ZERO⌈SS-240×PIX◊OPENPICTURE SCREEN◊→B16
[51]  L4:→(MD≤SS+PIX×360)/B15◊SS←(SS+240×PIX)⌊MD+(60|-MD)-360×PIX
[52]  OPENPICTURE SCREEN◊→B16
[53]  L5:K←SETPIX◊→K/B10◊OPENPICTURE SCREEN◊→(ZERO=ρJS)/B16◊→B17
[54]  L6:RESENL L-15◊→B11
```

```
L10:M1←,RECT[L;]⋄K←M1⋄PENPAT GRAYPAT⋄PENMODE TEN⋄TEXTFACE ZERO
B18:→(~BUTTON)/B19⋄FRAMERECT M1⋄G1←0 24 60⊤IZρSS+PIX×M1[TWO]-15
MOVETO 20 460⋄DRAWTEXT (NTWO↑⍕G1[TWO]),':',NTWO↑⍕100+NONE↑G1
M1←K+FOURρZERO,ZERO⌊¯385⌈ONE↓GETMOUSE-M⋄FRAMERECT M1⋄→B18
B19:FRAMERECT 2 4ρK,M1⋄PENPAT BLACKPAT⋄PENMODE EIGHT⋄→(375≤M1[TWO])/B15
G2←SS+PIX×M1[TWO]-15⋄K←T2[JS]⋄→(G2≤K)/B46
ERRORWINDOW 'Start Time Is After Latest Start Time Of ',,CVTTD K⋄→B10
B46:L←,L-19⋄G6←PDS+G2⋄KK←NTWO+(SP1>G2)⍳ONE
G4←KK↓SP1⋄G1←KK+⍳NONE+(G6[L]>G4)⍳ZERO⋄→(ZERO≠+/TL[G1])/B15⋄→(ZERO=CS)/B26
→(ZERO≠+/SPA[G1;L])/B15⋄→(CS=ONE)/B26⋄G3←(PDS≥ZERO)/ICL⋄K←ONE
B20:G5←G3[K]⋄→(G5=L)/B21⋄G1←KK+⍳NONE+(G6[G5]>G4)⍳ZERO
→(ZERO≠+/SPA[G1;G5]-TL[G1])/B22
B21:K←K+ONE⋄→(K≤ρG3)/B20⋄→(CS≤ρG3)/B23
B44:K←'There Are Not ',(⍕CS),' Crewmembers Available At This Time'
ERRORWINDOW K⋄→B10
B22:G3←G3 ELIM K⋄→(K≤ρG3)/B20⋄→(CS>ρG3)/B44
B23:G1←SR2[G3;]⋄G1[;TWO]←M1[TWO]⋄G1[;FOUR]←376⌊⌊15.5+(G6[G3]-SS)+PIX
STR[JS]←G2⋄→(CS=ρG3)/B24⋄G5←G3⍳L⋄STRIPEPAT FILLRECT G1
GRAYPAT FILLRECT G1[G5;]⋄FRAMERECT G1⋄L←(G1 SELWINDOW G5,CS)/G3⋄→B28
B24:L←G3⋄→(G2>SS+360×PIX)/B28⋄GRAYPAT FILLRECT G1⋄FRAMERECT G1⋄K←ONE
B25:→(G6[L[K]]≤SS)/B43⋄MOVETO ¯2 ¯2+G1[K;1 2]⋄DRAWTEXT 'J',⍕JS
B43:K←K+ONE⋄→(K≤CS)/B25⋄→B28
B26:M1[FOUR]←M1[FOUR]⌊376⋄STR[JS]←G2
B27:TEXTFACE ZERO⋄→(G2>SS+360×PIX)/B28⋄→(G6[L]<SS)/B28⋄GRAYPAT FILLRECT M1
FRAMERECT M1⋄→(G6[L]≤SS)/B28⋄MOVETO ¯2 ¯2+TWO↑M1⋄DRAWTEXT 'J',⍕JS
B28:PR[JS;]←ZERO⋄PR[JS;(CS≠ZERO)/L]←ONE⋄L ASSIGN G2,G6⋄→AJS/B42⋄→B11
L11:JOBC L⋄→(L=ONE)/B11⋄→(∨/L=3 8)/B10⋄→TW/B11⋄B DRAWPICTURE JW
JW←CUTPICTURE B⋄→B11
L12:→(ZERO=ρJS)/B10⋄L←IZρL-27⋄→(L>CL)/B10⋄G6←PDS[L]⋄→(G6<ZERO)/B15
→(CS=ZERO)/B15⋄→(CS≠ONE)/B30⋄KK←ZERO=SPA[;L]-TL⋄KK←(KK≠ZERO,NONE↓KK)/SP1
G1←⌊PFIVE×ρKK⋄KK←(G1,TWO)ρKK⋄G4←((-G6)≥-/KK)⍳ONE⋄→(G4≤G1)/B29
B45:ERRORWINDOW 'I Am Unable To Schedule This Crewmember'⋄→B10
B29:G2←KK[G4;ONE]⋄→(G2>T2[JS])/B45⋄G6←PDS+G2⋄M1←SR2[L;]
M1[TWO]←⌊15.5+ZERO⌈(G2-SS)+PIX⋄M1[FOUR]←376⌊⌊15.5+(G6[L]-SS)+PIX⋄L←,L
20 460 DRAWTIME G2⋄→B27
B30:→(CS>+/PDS>ZERO)/B51⋄L←L SELWINDOW2 CS
G2←PDS[L] ISEC ZERO=SPA[;L]+⍉(CS,ρTL)ρTL⋄→(G2≤T2[JS])/B31
B51:ERRORWINDOW 'I Am Unable To Schedule These Crewmembers'⋄→B10
B31:G6←G2+PDS⋄G1←SR2[L;]⋄G1[;TWO]←⌊15.5+ZERO⌈(G2-SS)+PIX
G1[;FOUR]←376⌊⌊15.5+ZERO⌈(G6[L]-SS)+PIX⋄G3←L⋄20 460 DRAWTIME G2⋄→B24
L13:→(ZERO=ρJS)/B10⋄→(CS=ZERO)/A5⋄→B15
A0:→(M[TWO]=OPTIONSN,PLANN,CONSTN,MOUSEN)/B32,B5,B33,A13⋄→B11
B32:→AA[M[THREE]]
B5:→AB[M[THREE]]
B33:→(M[THREE]=1 2 3)/A11,A12,A20
A1:JOBC 12⋄→B10
A2:→AJS/B41⋄AJS←ONE⋄PLANMENU[TWO;TWO]←⎕TCDC2⋄PLANN SETMENU PLANMENU
→(ZERO≠ρ,JS)/B11
B42:→TW/B40⋄B DRAWPICTURE SW
B40:AJOB⋄→TW/B11⋄TW←ONE⋄JOBC 12⋄→B10
B41:AJS←ZERO⋄PLANMENU[TWO;TWO]←' '⋄PLANN SETMENU PLANMENU⋄→B11
A3:→(ZERO≠ρ,JS)/B11⋄→TW/B51⋄B DRAWPICTURE SW
B51:AJOB2⋄→TW/B11⋄TW←ONE⋄JOBC 12⋄→B10
A4:ACS←~ACS⋄PLANMENU[FOUR;TWO]←(' ',⎕TCDC2)[ONE+ACS]
PLANN SETMENU PLANMENU⋄→(ACS∧AJS)/B42
A5:→(ZERO=ρ,JS)/B11
B34:→TW/B35⋄B DRAWPICTURE SW
B35:CSCHED⋄→(ZERO≠ρ,JS)/B10⋄→AJS/B42⋄→B11
A14:→TW/B52⋄JW←CUTPICTURE B⋄B DRAWPICTURE SW
B52:→(ZERO=ρJS)/B53⋄ERASERECT BR⋄GRAYPAT FILLRECT M4⋄FRAMERECT M4⋄JS←IZ
ERASERECT 3 4ρ207 370 218 510 25 438 33 510 32 378 168 510⋄RECT[19+ICL;]←0
B53:AUTOPLAN⋄→B10
A6:KK←⎕GETBITS LOADRECT⋄K←'Enter File Name:' FILESAVE FN
KK ⎕PUTBITS LOADRECT⋄→(ZERO=ρK)/B11⋄FN←(ONE-(⌽K)⍳':')↑K⋄G1←⎕ELX⋄⎕ELX←'→B36'
```

```
[119] K←K DATASAVE SCHEDDATA◇⎕ELX←G1◇ERRORWINDOW 'SAVE COMPLETED'◇→B11
[120] B36:ERRORWINDOW 'DISK/WORKSPACE FULL:  SAVE NOT COMPLETED -- FILE DELETED'
[121] ⎕ELX←'→B38'◇FN ⎕FERASE ⌈/⎕FNUMS
[122] B38:⎕ELX←G1◇→B11
[123] A7:KK←⎕GETBITS LOADRECT◇K←FILEOPEN 'SCHED_'◇KK ⎕PUTBITS LOADRECT
[124] →(∧/ZERO=(ρR),ρK)/A10◇→(ZERO=ρK)/B11◇FN←(ONE-(⌽K)⍳':')↑K
[125] DATALOAD K◇HEURISTICS[;ONE]←' '◇HEURISTICS[HEU;ONE]←'*'◇PUTMENUS◇→B1
[126] A8:KK←⎕GETBITS LOADRECT◇K←FILEOPEN 'SCHED_'◇KK ⎕PUTBITS LOADRECT
[127] →(ZERO=ρK)/B11◇G1←ONE+⌈/ZERO,⎕FNUMS
[128] K ⎕FHTIE G1◇K ⎕FERASE G1◇ERRORWINDOW 'FILE DELETED'◇→B11
[129] A9:→TW/B50◇B DRAWPICTURE SW
[130] B50:PRTSCHED◇→B10
[131] A10:ERASERECT SCREEN◇⎕CP←0 0◇T◇→ZERO
[132] A11:SETCON1◇→B11
[133] A12:DELETEMENUS◇K←(JL,ONE)ρ' '◇KK←JSS,JS◇K[KK;]←'*'◇K←K,JON◇G1←MSG
[134] MSG←' Select Job '◇K←K CHOICE2 KK◇MSG←G1◇→(ZERO=ρK)/B37◇K←SETTAR K
[135] B37:PUTMENUS◇→B11
[136] A13:MOUSEMENU[MODE+ONE;TWO]←' '◇MODE←M[THREE]
[137] MOUSEMENU[MODE+ONE;TWO]←⎕TCDC2◇MOUSEN SETMENU MOUSEMENU◇→B11
[138] A15:DELETEMENUS◇KK←MSG◇MSG←' Select Heuristic '◇K←HEURISTICS CHOICE2 IZ
[139] PUTMENUS◇MSG←KK◇→(ZERO=ρK)/B11◇HEURISTICS[HEU,K;ONE]←' *'◇HEU←K◇→B11
[140] A16:K←SETSEARCH◇→B11
[141] A20:DELETEMENUS◇PRIORITYGET◇PUTMENUS◇→B11
    ∇


    ∇INIT1 R;E;TL;JT;SR2;M1;MSG;PDS;SPA;JSS;BR;WL;LC;RC;TC;BC;OUT;RE2;G6;PRC
[1]   INIT2 R
    ∇


    ∇INIT2 R;S1;S2;S3;S4;S5;S6;SIZE;ULC;D;SW;JW;TW;ACS;CS;KK;FL10;WTEXT;AA;RE;B
[1]   INIT3 R
    ∇


    ∇INIT3 R;FN;JL;IJL;MPD;MPE;TC0;T2;T3;TAR;MR;AJS;TT;ST1;MPF;MPG;STR;AP;OC;AB
[1]   INIT4 R
    ∇


    ∇INIT4 R;CL4;CL5;CL6;AUDX;RESX;RESM;SP1;M4;P1;MODE;ST;PR;CT;PRI;FL8;HEU;NEC
[1]   INIT R
    ∇


   ∇INITCURSOR∇


   ∇INVERTRECT∇


    ∇IR2A;SROW;G1
[1]   SROW←⌈(M[ONE]-ONE+S1)÷11◇→(SROW<ONE)/ZERO◇→(SROW>BC)/ZERO
[2]   →(THREE≥SROW+TC)/ZERO
[3]   →((SROW+TC)>(ρWTEXT)[ONE])/ZERO◇→(~(SROW+TC-THREE)∊JSS)/B12
[4]   ERRORWINDOW 'THIS JOB HAS ALREADY BEEN SCHEDULED'◇→ZERO
[5]   B12:INVERTRECT D+(SROW×11),ZERO,(SROW×11),SIX×RC◇→(ZERO=ρJS)/B7
[6]   B1:→(JR=SROW+TC)/B3◇→((JR≤TC)∨JR>TC+BC)/B2◇G1←11×JR-TC
```

```
[7]   INVERTRECT D+G1,ZERO,G1,SIX×RC
[8]   B2:JW←CUTPICTURE B◇OPENPICTURE SCREEN◇B DRAWPICTURE SW◇SPA←SP◇PRS 1 1
[9]   ERASERECT 3 4ρ207 370 218 510 25 438 33 510 32 378 168 510
[10]  JS←IZρSROW+TC-THREE◇→B5
[11]  B3:JW←CUTPICTURE B◇OPENPICTURE SCREEN◇B DRAWPICTURE SW◇SPA←SP◇ERASERECT BR
[12]  GRAYPAT FILLRECT M4◇FRAMERECT M4◇JS←IZ
[13]  ERASERECT 3 4ρ207 370 218 510 25 438 33 510 32 378 168 510
[14]  SCREEN DRAWPICTURE CLOSEPICTURE◇RECT[19+ICL;]←ZERO
[15]  →TW/ZERO◇SW←CUTPICTURE B◇B DRAWPICTURE JW◇→ZERO
[16]  B7:JS←IZρSROW+TC-THREE◇JW←CUTPICTURE B◇OPENPICTURE SCREEN◇B DRAWPICTURE SW
[17]  B5:SETSCHED
[18]  →ACS/ZERO◇→TW/ZERO◇SW←CUTPICTURE B◇B DRAWPICTURE JW◇→ZERO
    ∇


    ∇IRAUTO;SROW;K;G1;G2;SCOL;G;F
[1]   G←IZ
[2]   B4:→(ZERO=ρAUTV)/B2◇SCOL←ONE↑AUTV◇SROW←AUTV[TWO]◇G←G,SROW◇AUTV←TWO↓AUTV
[3]   B1:→(L1,L2,L3,L4,L5,L6,L7)[SCOL]
[4]   L1:K←ONE↑AUTV◇AUTV←ONE↓AUTV◇CONP[ONE+SROW;ONE]←-K◇CT[1+SROW;ONE]←-K◇→B4
[5]   B2:CONP←CONP⌊CONP[;ONE]∘.+CONP[ONE;]◇CT←CT⌊CT[;ONE]∘.+CT[ONE;]
[6]   G←(IJL∊G)/IJL
[7]   CONP←CONP⌊CONP[;ONE+SROW]∘.+CONP[1+SROW;]◇CT←CT⌊CT[;1+SROW]∘.+CT[1+SROW;]
[8]   B30:→(ZERO=ρG)/B5◇K←G[ONE]◇G←ONE↓G◇CONP←CONP⌊CONP[;ONE+K]∘.+CONP[ONE+K;]
[9]   CT←CT⌊CT[;ONE+K]∘.+CT[ONE+K;]◇→B30
[10]  B5:T2←TCLST IJL◇T3←T3⌊T2+MPE◇K←ONE
[11]  B13:T3[K]←⌊/T3[K],T2[-(PRCιK)↓(NONE+PRCιK+ONE)↑PRC]◇K←K+ONE◇→(K≤JL)/B13
[12]  →ZERO
[13]  L2:K←ONE↑AUTV◇AUTV←ONE↓AUTV◇CONP[ONE;ONE+SROW]←K◇CT[ONE;ONE+SROW]←K◇→B4
[14]  L3:K←ONE↑AUTV◇AUTV←ONE↓AUTV◇T3[SROW]←K◇TT[SROW]←K
[15]  →(K≥CONP[ONE;ONE+SROW]+MPD[SROW])/B20◇CONP[ONE;ONE+SROW]←K-MPD[SROW]
[16]  CT[ONE;ONE+SROW]←K-MPD[SROW]◇→B4
[17]  B20:→(K≥CT[ONE;ONE+SROW]+MPD[SROW])/B4◇CT[ONE;ONE+SROW]←K-MPD[SROW]◇→B4
[18]  L4:F←ONE↑AUTV◇K←AUTV[TWO]◇CONP[ONE+SROW;ONE+F]←-K◇CT[ONE+SROW;ONE+F]←-K
[19]  B6:G←G,F◇AUTV←TWO↓AUTV◇→B4
[20]  L5:F←ONE↑AUTV◇K←AUTV[TWO]◇CONP[ONE+F;ONE+SROW]←K◇CT[ONE+F;ONE+SROW]←K◇→B6
[21]  L6:F←ONE↑AUTV◇K←AUTV[TWO]◇→(K=TWO)/B10◇G1←PRCιSROW◇PRC←(G1↑PRC),(-F),G1↓PRC
[22]  →(CONP[ONE+F;ONE+SROW]≤-MPD[SROW])/B6
[23]  CONP[ONE+F;ONE+SROW]←-MPD[SROW]◇CT[ONE+F;ONE+SROW]←-MPD[SROW]◇→B6
[24]  B10:G2←PRCιF◇PRC←(G2↑PRC),(-SROW),G2↓PRC
[25]  →((-CONP[ONE+SROW;ONE+F])≥MPD[F])/B6
[26]  CONP[ONE+SROW;ONE+F]←-MPD[F]◇CT[ONE+SROW;ONE+F]←-MPD[F]◇→B6
[27]  L7:F←ONE↑AUTV◇CONP[ONE+SROW;ONE+F]←ZERO◇CONP[ONE+F;ONE+SROW]←ZERO
[28]  CT[ONE+SROW;ONE+F]←ZERO◇CT[ONE+F;ONE+SROW]←ZERO◇→B6
    ∇


    ∇IRC;SROW;K;G1;G2;A;B;SCOL;DEF;H;OPT
[1]   SROW←⌈(M[ONE]-ONE+S1)÷11◇→(SROW<ONE)/ZERO◇→(TWO≥SROW+TC)/ZERO
[2]   →(SROW>BC)/ZERO
[3]   →((SROW+TC)>TWO+JL)/ZERO◇K←ONE⌈⌈(M[TWO]-S2+TEN+SIX×(ρTC0)[TWO])÷SIX
[4]   H←+\AV125=WT[ONE;]◇SCOL←1+H[K+LC]◇G1←6×(ρTC0)[2]+0⌈((SCOL≠1)×HιSCOL-1)-LC
[5]   G2←(S4-S2+NINE)⌊ONE+SIX×(ρTC0)[TWO]+(HιSCOL)-LC+ONE
[6]   INVERTRECT D+(11×SROW),G1,(11×SROW),G2◇→(ZERO=ρ,JS)/B15
[7]   ERRORWINDOW 'DATA CANNOT BE MODIFIED WHILE A JOB IS SELECTED'◇→ZERO
[8]   B15:SROW←IZρSROW+TC-TWO◇→(SCOL<FOUR)/B1◇→(~∨/(ONE↑F)=ZERO,SROW)/B1
[9]   ERRORWINDOW 'THIS DATA CANNOT BE MODIFIED'◇→ZERO
[10]  B1:→(L1,L2,L3,L4,L5,L6,L7,L8)[SCOL]
[11]  L1:DEF←T1[SROW]
[12]  K←(EIGHT,DEF,DEF,T2[SROW]) REPLYWINDOW 'ENTER NEW EARLIEST START TIME:'
[13]  →(OPT=2 3 6)/0 0,B21◇→(K=DEF)/ZERO◇CONP[ONE+SROW;ONE]←-K◇CT[1+SROW;ONE]←-K
[14]  B2:AUTV←AUTV,SCOL,SROW,K
```

```
[15]  CONP←CONP⌊CONP[;ONE]∘.+CONP[ONE;]⋄CT←CT⌊CT[;ONE]∘.+CT[ONE;]
[16]  CONP←CONP⌊CONP[;ONE+SROW]∘.+CONP[1+SROW;]⋄CT←CT⌊CT[;1+SROW]∘.+CT[1+SROW;]
[17]  B5:T1←TCEST IJL⋄TC1←(HE1 CAPPEND CVTTD T1),AV125⋄T2←TCLST IJL
[18]  TC2←(HE2 CAPPEND CVTTD T2),AV125⋄T3←T3⌊T2+MPE
[19]  →(ZERO=⍴F)/B12⋄W4←,TCMIN F⋄W5←,TCMAX F⋄W7←(JL,THREE)⍴'YES'
[20]  TC4←(HE4 CAPPEND CVTTD W4),AV125⋄TC5←(HE5 CAPPEND CVTTD W5),AV125
[21]  W7[(,(W5≠ZERO)∨W5≠W4)/IJL;]←' '⋄TC7←(HE7 CAPPEND W7),AV125
[22]  B12:K←ONE
[23]  B13:T3[K]←⌊/T3[K],T2[-(PRC⍳K)↓(NONE+PRC⍳K+ONE)↑PRC]⋄K←K+ONE⋄→(K≤JL)/B13
[24]  TC3←(HE3 CAPPEND CVTTD T3),AV125
[25]  B3:WT←TC1,TC2,TC3,TC4,TC5,TC6,TC7,TC8
[26]  WT[TWO;]←'='⋄WT1←(ONE↓⍴WT)+ONE↓⍴TC0⋄→ZERO
[27]  B21:HELPWINDOW CONHELP1⋄→L1
[28]  L2:DEF←T2[SROW]⋄K←'ENTER NEW LATEST START TIME:'
[29]  K←(EIGHT,DEF,T1[SROW],DEF) REPLYWINDOW K⋄→(OPT=2 3 6)/0 0,B22
[30]  →(K=DEF)/ZERO⋄CONP[ONE;ONE+SROW]←K⋄CT[ONE;ONE+SROW]←K⋄→B2
[31]  B22:HELPWINDOW CONHELP2⋄→L2
[32]  L3:DEF←T3[SROW]⋄G1←'ENTER NEW LATEST END TIME:'
[33]  K←(EIGHT,DEF,(T1[SROW]+MPD[SROW]),DEF) REPLYWINDOW G1⋄→(OPT=2 3 6)/0 0,B23
[34]  →(K=DEF)/ZERO⋄T3[SROW]←K⋄TT[SROW]←K⋄TC3←(HE3 CAPPEND CVTTD T3),AV125
[35]  →(K≥T2[SROW]+MPD[SROW])/B20⋄CONP[ONE;ONE+SROW]←K-MPD[SROW]
[36]  CT[ONE;ONE+SROW]←K-MPD[SROW]⋄→B2
[37]  B20:AUTV←AUTV,SCOL,SROW,K
[38]  →(K≥CT[ONE;ONE+SROW]+MPD[SROW])/B3⋄CT[ONE;ONE+SROW]←K-MPD[SROW]
[39]  CT←CT⌊CT[;ONE]∘.+CT[ONE;]⋄CT←CT⌊CT[;ONE+SROW]∘.+CT[ONE+SROW;]⋄→B3
[40]  B23:HELPWINDOW CONHELP3⋄→L3
[41]  L4:DEF←W4[SROW]⋄K←'ENTER NEW DELTA-MIN:'
[42]  K←(EIGHT,DEF,DEF,W5[SROW]) REPLYWINDOW K⋄→(OPT=2 3 6)/0 0,B24
[43]  →(K=DEF)/ZERO⋄CONP[ONE+SROW;ONE+F]←-K⋄CT[ONE+SROW;ONE+F]←-K
[44]  B6:CONP←CONP⌊CONP[;IZ⍴1+F]∘.+CONP[IZ⍴1+F;]⋄CT←CT⌊CT[;IZ⍴1+F]∘.+CT[IZ⍴1+F;]
[45]  CONP←CONP⌊CONP[;ONE+SROW]∘.+CONP[ONE+SROW;]
[46]  CT←CT⌊CT[;ONE+SROW]∘.+CT[ONE+SROW;]⋄AUTV←AUTV,SCOL,SROW,F,K⋄→B5
[47]  B24:HELPWINDOW CONHELP4⋄→L4
[48]  L5:DEF←W5[SROW]⋄K←'ENTER NEW DELTA-MAX:'
[49]  K←(EIGHT,DEF,W4[SROW],DEF) REPLYWINDOW K⋄→(OPT=2 3 6)/0 0,B25
[50]  →(K=DEF)/ZERO⋄CONP[ONE+F;ONE+SROW]←K⋄CT[ONE+F;ONE+SROW]←K⋄→B6
[51]  B25:HELPWINDOW CONHELP5⋄→L5
[52]  L6:K←' '=W6[SROW;ONE]
[53]  G1←MPD[SROW]≤-W4[SROW]⋄G2←MPD[F]≤W5[SROW]
[54]  K←(K×G2+2×G1) BAWINDOW 'SELECT BEFORE OR AFTER:'⋄→(K=ZERO)/ZERO
[55]  →(K=TWO)/B10⋄G1←PRC⍳SROW⋄PRC←(G1↑PRC),(-F),G1↓PRC⋄W6[SROW;]←'BEFORE'
[56]  TC6←(HE6 CAPPEND W6),AV125⋄→(W5[SROW]≤-MPD[SROW])/B30
[57]  CONP[ONE+F;ONE+SROW]←-MPD[SROW]⋄CT[ONE+F;ONE+SROW]←-MPD[SROW]⋄→B6
[58]  B10:G2←PRC⍳F⋄PRC←(G2↑PRC),(-SROW),G2↓PRC⋄W6[SROW;]←'AFTER '
[59]  TC6←(HE6 CAPPEND W6),AV125⋄→(W4[SROW]≥MPD[F])/B30
[60]  CONP[ONE+SROW;ONE+F]←-MPD[F]⋄CT[ONE+SROW;ONE+F]←-MPD[F]⋄→B6
[61]  B30:AUTV←AUTV,SCOL,SROW,F,K⋄→B12
[62]  L7:→(W7[SROW;ONE]=' ')/B7⋄ERRORWINDOW 'THIS CONSTRAINT CANNOT BE RELAXED'
[63]  →ZERO
[64]  B7:→(W4[SROW]≤ZERO)/B8⋄ERRORWINDOW 'DELTA-MIN VIOLATION'⋄→ZERO
[65]  B8:→(W5[SROW]≥ZERO)/B9⋄ERRORWINDOW 'DELTA-MAX VIOLATION'⋄→ZERO
[66]  B9:CONP[ONE+SROW;ONE+F]←ZERO⋄CONP[ONE+F;ONE+SROW]←ZERO
[67]  CT[ONE+SROW;ONE+F]←ZERO⋄CT[ONE+F;ONE+SROW]←ZERO⋄K←ZERO⋄→B6
[68]  L8:G1←NEC⍳F⋄G2←-G1↓(NONE+NEC⍳F+ONE)↑NEC⋄→(SROW∊G2)/B30
[69]  TC8[TWO+SROW;2 3 4]←'NEC'⋄NEC←(G1↑NEC),(-SROW),G1↓NEC⋄→B3
[70]  B30:TC8[TWO+SROW;2 3 4]←'   '
[71]  NEC←(G1↑NEC),(-(G2≠SROW)/G2),(NONE+NEC⍳ONE+F)↓NEC⋄→B3
    ∇


    ∇IRCON;SROW;K;G1;G2;A;B
[1]   SROW←⌈(M[ONE]-ONE+S1)+11⋄→(SROW<ONE)/ZERO⋄→(TWO≥SROW+TC)/ZERO
[2]   →((SROW+TC)>TWO+JL)/ZERO⋄→(SROW>BC)/ZERO
```

```
◇→(ZERO=ρF)/B5◇→(F≠SROW+TC-TWO)/B5◇F←IZ◇K←(TWO+JL),SIX◇TC8←K↑HE8
TC4←(K↑HE4),AV125◇TC5←(K↑HE5),AV125◇TC6←(K↑HE6),AV125◇TC7←(K↑HE7),AV125
B1:WT←TC1,TC2,TC3,TC4,TC5,TC6,TC7,TC8
WT[TWO;]←'='◇WT1←(ρWT)[TWO]+(ρTC0)[2]◇→ZERO
B5:F←,SROW+TC-?WO◇W4←,TCMIN F◇W5←,TCMAX F◇W7←(JL,THREE)ρ'YES'
TC4←(HE4 CAPPEND CVTTD W4),AV125◇TC5←(HE5 CAPPEND CVTTD W5),AV125
W7[((W5≠ZERO)∨W5≠W4)/IJL;]←'  '◇TC7←(HE7 CAPPEND W7),AV125
A←-(PRC⍳F)↓(NONE+PRC⍳F+ONE)↑PRC◇B←(PRC=-F)/+\PRC>ZERO
W6←(JL,SIX)ρ' '◇W6[A;]←((ρA),SIX)ρ'AFTER '◇W6[B;]←((ρB),SIX)ρ'BEFORE'
TC6←(HE6 CAPPEND W6),AV125◇A←(TWO+F),TWO-(NEC⍳F)↓(NONE+NEC⍳F+ONE)↑NEC
TC8←((TWO+JL),SIX)↑HE8◇TC8[A;2 3 4]←((ρA),THREE)ρ'NEC'◇→B1
∇


∇R←VAL ISEC SPB;V;G1;G2;K;KK;J;G4;G3;N
V←∧/SPB◇N←⌊/VAL◇R←MD◇G1←(V≠ZERO,NONE↓V)/SP1◇G2←⌊PFIVE×ρG1◇G1←(G2,TWO)ρG1
→(∧/VAL=N)/B1◇G1←((-N)≥-/G1)⌿G1◇KK←(ρG1)[ONE]◇→(KK=ZERO)/ZERO◇K←ONE
B5:G4←G1[K;TWO]◇G3←G1[K;ONE]◇→((G4-G3)>⌈/VAL)/B7◇J←ONE◇→(VAL[J]≤G4-G3)/B2
B3:→(~∧/((SP1≥G4)∧SP1<G3+VAL[J])/SPB[;J])/B4
B2:J←J+ONE◇→(J≤CS)/B3
B7:R←G3◇→ZERO
B4:K←K+ONE◇→(K≤KK)/B5◇→ZERO
B1:K←((-N)≥-/G1)⍳ONE◇→(K>G2)/ZERO◇R←G1[K;ONE]
∇


∇JOBATT H;UN;UN2;K;KK;REC;M;J;L;IR;N;G1;G2;G3;G4;W1;G5;P;G6;SR;NN;PDS;CS
DELETEMENUS◇K←ONE◇SR←STR[H]◇W1←SP[;CL6]◇G4←ST[H]◇G1←W1⍳G4◇N←P1[H;]
CN←N/ICL◇JS←IZ◇CS←JT[H]◇G2←(NONE+G1)+⍳(W1⍳ST1[H])-G1◇G1←SP◇G5←SP[G2;CN]
G5←G5×G5≠H◇SP[G2;CN]←G5◇SP[G2;CL4]←SP[G2;CL4]-((ρG2),FOUR)ρRESM[H;]
SP[G2;CL5]←ONE⌊NONE⌈+/(ZERO,AUDX)[ONE+SP[G2;ICL]]◇G2←CONP◇CONP←CT◇G3←T3
T3←TT◇G6←MPE[H]◇MPE[H]←MPG[H]◇G5←MPD[H]◇MPD[H]←MPF[H]
PDS←(CL⌈EIGHT×CS=ZERO)ρPD[H;]
B7:G4←JSS[K]◇→(G4=H)/B4◇NN←G4+ONE◇KK←ONE-(PRC⍳G4)↓(NONE+PRC⍳NN)↑PRC
B5:→(ZERO=ρKK)/B6◇P←KK[ONE]◇CONP[P;NN]←-MPD[G4]
CONP←CONP⌊CONP[;P]∘.+CONP[P;]◇KK←ONE↓KK◇→B5
B6:CONP[ONE;NN]←ST[G4]◇CONP[NN;ONE]←-ST[G4]◇T3[G4]←ST1[G4]
CONP←CONP⌊CONP[;ONE]∘.+CONP[ONE;]◇CONP←CONP⌊CONP[;NN]∘.+CONP[NN;]
B4:K←K+ONE◇→(K≤ρJSS)/B7◇G4←CONP[ONE;ONE+IJL]◇T3←T3⌊G4+MPE◇K←ONE
B8:T3[K]←⌊/T3[K],G4[-(PRC⍳K)↓(NONE+PRC⍳K+ONE)↑PRC]◇K←K+ONE◇→(K≤JL)/B8
UN←68 50 276 466◇OPENPICTURE UN◇UN2←⎕GETBITS UN◇MOVETO 122 65
UN DRAWPICTURE JOBATTV
DRAWTEXT 'CREW REQUIRED = ',(⍕CS),'   JOB PRIORITY = ',⍕PRI[H]◇MOVETO 93 65
K←'EST =',(,CVTTD -CONP[ONE+H;ONE]),'   LST =',,CVTTD CONP[ONE;ONE+H]
DRAWTEXT K,'   LET =',,CVTTD T3[H]◇MOVETO 108 65
DRAWTEXT 'MINIMUM PERFORMANCE TIME = ',⍕MPD[H]◇MOVETO 165 65
DRAWTEXT CRN APPEND 'ZERO CREW JOBS'◇MOVETO 165 272◇KK←⍕((ONE+CL),ONE)ρPDS
→(CS=ZERO)/B25◇KK[((PDS<ZERO)/ICL),CL+ONE;]←' '
B26:DRAWTEXT KK◇MOVETO 165 337
DRAWTEXT ⍕((ONE+CL),ONE)ρWL,+/PD[(JT[JSS]=ZERO)/JSS;ONE]
MOVETO 165 402◇DRAWTEXT ⍕((ONE+CL),ONE)ρ(+/PR×PD),+/PD[(JT=ZERO)/IJL;ONE]
K←⍉(FOUR,CL)ρ(148+TEN×ICL),(CLρ205),(156+TEN×ICL),CLρ230
TEXTFACE ONE◇REC←ATTRECTS K◇IR←⍳FIVE+CL◇J←L1,L2,L3,L4,L5,CLρL6
N[PR[H;]/ICL]←TWO◇GRAYPAT FILLRECT (N=ONE)⌿K◇BLACKPAT FILLRECT (N=TWO)⌿K
FRAMERECT K◇MOVETO 83 65◇DRAWTEXT 'JOB = ',JON[H;]◇MOVETO 113 270
DRAWTEXT 'START TIME ='
B12:K←STR[H]⌈(STR[H]<ZERO)×ST[H]◇KK←IZ◇→(K<ZERO)/B1◇KK←,CVTTD K
B1:MOVETO 113 378◇DRAWTEXT KK◇→(STR[H]<ZERO)/B10◇INVERTRECT 104 376 115 447
B10:TEXTFACE ZERO◇UN DRAWPICTURE CLOSEPICTURE
B2:M←⎕GETKEY◇→(THREE≠ρM)/B2◇→(ONE≠M[ONE])/B2◇M←M[2 3]
L←ONE↑(M PTINRECT REC)/IR◇→(ZERO=L)/B2◇→J[L]
B25:KK[ICL;]←' '◇→B26
```

```
[36]  L1:STR[H]←SR◊SP←G1◊CONP←G2◊T3←G3◊UN2 ⎕PUTBITS UN◊MPD[H]←G5◊MPE[H]←G6
[37]  B30:PUTMENUS◊→ZERO
[38]  L2:ST[H]←NONE◊ST1[H]←NONE◊STR[H]←NONE◊T2←CONP[ONE;ONE+IJL]◊JSS←(JSS≠H)/JSS
[39]  WL[CN]←WL[CN]-PDS[CN]◊WTEXT[H+THREE;ONE]←' '◊P1[H;]←ZERO◊PR[H;]←ZERO
[40]  OPENPICTURE SCREEN◊UN2 ⎕PUTBITS UN◊PRS ZERO◊MOVETO 176 460◊TEXTFACE ONE
[41]  DRAWTEXT EIGHT↑VCEN2 +/PRI[JSS]◊TEXTFACE ZERO
[42]  SCREEN DRAWPICTURE CLOSEPICTURE◊→TW/B30◊OPENPICTURE B◊B DRAWPICTURE JW
[43]  ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◊MOVETO 10 8+S1,S2
[44]  DRAWTEXT (BC,RC)↑(TC,LC)↓WTEXT◊JW←CLOSEPICTURE◊→B30
[45]  L3:G4←ST[H],ST1[H],P1[H;],PR[H;]◊P←N=TWO◊→(CS<+/P)/B14
[46]  NN←⎕GETBITS UN◊UN2 ⎕PUTBITS UN◊PR[H;]←P◊P1[H;]←ZERO
[47]  B21:JS←H◊CONPROC◊M←TW◊TW←ONE◊WL[CN]←WL[CN]-PDS[CN]◊ST[H]←NONE◊ST1[H]←NONE
[48]  CSCHED◊TW←M◊WL[CN]←WL[CN]+PDS[CN]◊→(ZERO=ρ,JS)/B22◊JS←IZ◊P1[H;]←CL↑TWO↓G4
[49]  PR[H;]←(TWO+CL)↓G4◊ST[H]←G4[ONE]◊ST1[H]←G4[TWO]◊NN ⎕PUTBITS UN◊→B2
[50]  B14:ERRORWINDOW 'Too Many Crewmembers Have Been Selected'◊→B2
[51]  B22:WL[CN]←WL[CN]-PDS[CN]◊JSS←NONE↓JSS◊→B30
[52]  L4:P←N=TWO◊→(CS<+/P)/B14◊UN2 ⎕PUTBITS UN◊PR[H;]←P◊AUTOPLAN◊→B30
[53]  L5:INVERTRECT 104 376 115 447◊K←8 ¯1,(-CONP[ONE+H;ONE]),CONP[ONE;ONE+H]
[54]  K←K REPLYWINDOW 'ENTER START TIME:'◊→(OPT=2 3 6)/B11,B9,B31◊STR[H]←K
[55]  B13:TEXTFACE ONE◊OPENPICTURE UN◊ERASERECT 104 376 115 447◊→B12
[56]  B11:INVERTRECT 104 376 115 447◊→B2
[57]  B9:STR[H]←NONE◊→B13
[58]  B31:HELPWINDOW JOBATTHELP◊→B11
[59]  L6:L←IZρL-FIVE◊→((CS=ZERO)∨PDS[L]<ZERO)/B3
[60]  N[L]←(TWO×N[L]≠TWO)+(L∊CN)×N[L]=TWO
[61]  PATS[N[L]+ONE;] FILLRECT REC[L+FIVE;]+1 1 ¯1 ¯1◊→B2
[62]  B3:⎕SOUND BEEP◊→B2
    ∇


    ∇JOBATT1 H;JS;CN;OPT
[1]   JOBATT H
    ∇


    ∇JOBC L;G1;G2;K;G3;G4;G5;TR;M1;G5;A;CR;HS;JR
[1]   TEXTFACE ZERO◊JR←THREE+ONE⍴JS◊→(L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12)[L]
[2]   L11:S1←ULC[ONE]◊S2←ULC[TWO]◊S3←S1+SIZE[ONE]◊D←¯9 7 1 7+S1,S2,S1,S2
[3]   S4←S2+SIZE[TWO]◊BC←⌊(NTHREE+SIZE[ONE])÷11◊RC←⌊(NNINE+SIZE[TWO])÷SIX
[4]   OUT←(BC,RC)↑(TC,LC)↓WTEXT◊B←¯18 0 18 18+S1,S2,S3,S4◊RECT[ONE;]←S1,S2,S3,S4
[5]   SW←CUTPICTURE B◊OPENPICTURE B◊TR←¯18 0 1 18+S1,S2,S1,S4
[6]   ERASERECT B◊CR←¯14 10 ¯3 22+S1,S2,S1,S2◊G1←0 ¯1 18 18+S1,S4,S1,S4
[7]   G2←¯18 ¯1 0 18+S3,S4,S3,S4◊G2 DRAWPICTURE DOWNARROW
[8]   G1 DRAWPICTURE UPARROW◊G3←¯1 0 18 18+S3,S2,S3,S2◊G3 DRAWPICTURE LEFTARROW
[9]   G4←¯1 ¯18 18 0+S3,S4,S3,S4◊G4 DRAWPICTURE RIGHTARROW
[10]  G5←¯1 ¯1 18 18+S3,S4,S3,S4◊G5 DRAWPICTURE CORNER
[11]  K←(S1-15)+TWO×⍳SIX◊K←⍉4 6ρ(K-ONE),(SIXρTWO+S2),K,SIXρS4+16
[12]  RECT[ONE+⍳NINE;]←9 4ρCR,TR,G1,G2,G3,G4,G5◊FRAMERECT K⍪B⍪RECT[⍳EIGHT;]
[13]  G1←⌊PFIVE×18+S4+S2-MR⌊S4-S2+27◊K←(S1-17),G1,(S1-ONE),S2+18+S4-G1
[14]  ERASERECT K⍪3 4ρ(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
[15]  MOVETO (S1-THREE),G1◊TEXTSIZE 12◊TEXTFONT ZERO◊MSG DRAWMSG MR,MR⌊S4-S2+27
[16]  TEXTNORMAL
[17]  GRAYPAT FILLRECT 2 4ρ0 18 17 ¯18 18 0 ¯18 17+S3,S2,S3,S4,S1,S4,S3,S4
[18]  B1:MOVETO 10 8+S1,S2◊DRAWTEXT OUT◊→(ZERO=ρJS)/B2◊→(JR≤TC)/B2
[19]  →(JR>TC+BC)/B2◊INVERTRECT D+(11×JR-TC),ZERO,(11×JR-TC),SIX×RC
[20]  B2:→(L=10)/B4
[21]  B3:G1←(BC×¯36+S3-S1)+2×1↑ρWTEXT◊G2←S1+18+((TC+.5×BC)+1↑ρWTEXT)×(S3-S1)-36
[22]  S6←(⌊G2-G1),S4,((S3-18)⌊⌊G2+G1),17+S4◊ERASERECT S6◊FRAMERECT S6
[23]  RECT[NINE;]←S6◊→(L=4 5 9)/B5,B5,B5
[24]  B4:G5←(RC×¯36+S4-S2)+2×1↓ρWTEXT◊G2←S2+18+((LC+.5×RC)+1↓ρWTEXT)×(S4-S2)-36
[25]  S5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5◊ERASERECT S5◊FRAMERECT S5
[26]  RECT[TEN;]←S5
```

```
[27]  B5:JW←CLOSEPICTURE◊→ZERO
[28]  B6:⎕SOUND BEEP◊→ZERO
[29]  L1:→BUTTON/L1◊IR2A◊→ZERO
[30]  L2:→BUTTON/L2◊TW←ONE◊RECT[⍳TEN;]←ZERO
[31]  OPTIONS[TWO;TWO]←' '◊OPTIONSN SETMENU OPTIONS◊JW←IZ◊SW←IZ◊→ZERO
[32]  L3:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
[33]  B7:→(~BUTTON)/B8◊FRAMERECT M1◊M1←B+4⍴(0 0⌈GETMOUSE)-M◊FRAMERECT M1◊→B7
[34]  B8:ULC←ULC+(0 0⌈GETMOUSE)-M
[35]  B9:FRAMERECT M1◊PENPAT BLACKPAT◊B DRAWPICTURE SW◊PENMODE EIGHT◊→L11
[36]  L4:→(TC≤ZERO)/B6◊SCROLLRECT ((2 7,(2+11×BC),NTWO)+RECT[ONE;1 2 1 4]),0 11
[37]  OPENPICTURE B◊MOVETO 10 8+S1,S2◊DRAWTEXT RC↑LC↓,WTEXT[TC;]◊TC←TC-ONE
[38]  GRAYPAT FILLRECT S6◊→((JR=3)∨TC≠¯1+JR)/B3◊INVERTRECT D+11 0 11,SIX×RC◊→B3
[39]  L5:→(TC≥(⍴WTEXT)[ONE]-BC)/B6◊TC←TC+ONE
[40]  SCROLLRECT ((2 7,(TWO+11×BC),NTWO)+RECT[ONE;1 2 1 4]),0 ¯11
[41]  OPENPICTURE B◊MOVETO (S1+NONE+11×(⍴OUT)[ONE]),S2+EIGHT
[42]  DRAWTEXT RC↑LC↓,WTEXT[BC+TC;]◊GRAYPAT FILLRECT S6◊→((TC+BC)≠JR)/B3
[43]  INVERTRECT D+11 0 11 6×BC,ZERO,BC,RC◊→B3
[44]  L6:→(LC≤ZERO)/B6◊HS←(⌽(LC-ONE)↑WTEXT[ONE;])⍳AV125◊G3←LC⌊HS
[45]  LC←LC-G3◊OUT←(BC,G3⌊RC)↑(BC,RC)↑(TC,LC)↓WTEXT
[46]  SCROLLRECT ((1 7 ¯1,SEVEN+SIX×RC)+RECT[ONE;1 2 3 2]),(SIX×G3),ZERO
[47]  OPENPICTURE B◊MOVETO 10 8+S1,S2◊DRAWTEXT OUT◊GRAYPAT FILLRECT S5
[48]  →(ZERO=⍴JS)/B4◊→((JR≤TC)∨JR>TC+BC)/B4
[49]  M1←11×JR-TC◊INVERTRECT D+M1,ZERO,M1,ONE+SIX×G3◊→B4
[50]  L7:G3←(⍴WTEXT)[TWO]-RC◊→(LC≥G3)/B6◊HS←((RC+LC+ONE)↓WTEXT[ONE;])⍳AV125
[51]  G3←HS⌊G3◊SCROLLRECT ((1 7 ¯1,7+SIX×RC)+RECT[ONE;1 2 3 2]),(NSIX×G3),ZERO
[52]  G3←G3⌊RC◊LC←LC+G3◊OPENPICTURE B
[53]  OUT←(BC,-G3⌊RC)↑(BC,RC)↑(TC,LC)↓WTEXT◊MOVETO (S1+TEN),S2+EIGHT+SIX×RC-G3
[54]  DRAWTEXT OUT◊GRAYPAT FILLRECT S5◊→(ZERO=⍴JS)/B4◊→(JR≤TC)/B4
[55]  →(JR>TC+BC)/B4◊INVERTRECT D+11 6 11 6×(JR-TC),(RC-G3),(JR-TC),RC◊→B4
[56]  L8:M1←B◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[57]  B10:→(~BUTTON)/B11◊FRAMERECT M1
[58]  M1←B[1 2],(18 18+ULC)+36 36⌈SIZE+GETMOUSE-M◊FRAMERECT M1◊→B10
[59]  B11:SIZE←36 36⌈SIZE+GETMOUSE-M◊→B9
[60]  L9:M1←S6◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[61]  B12:→(~BUTTON)/B13◊FRAMERECT M1
[62]  K←(18+S1-S6[1])⌈(S3-18+S6[3])⌊1↑GETMOUSE-M◊M1←S6+4⍴K,0◊FRAMERECT M1◊→B12
[63]  B13:FRAMERECT M1◊OPENPICTURE B
[64]  GRAYPAT FILLRECT S6◊TC←TC+⌊PFIVE+K×(⍴WTEXT)[ONE]+S3-S1+36
[65]  OUT←(BC,RC)↑(TC,LC)↓WTEXT◊ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]
[66]  B14:PENPAT BLACKPAT◊PENMODE EIGHT◊→B1
[67]  L10:M1←S5◊PENPAT GRAYPAT◊PENMODE TEN◊K←ZERO
[68]  B15:→(~BUTTON)/B16◊FRAMERECT M1
[69]  K←(18+S2-S5[2])⌈(S4-18+S5[4])⌊1↓GETMOUSE-M◊M1←S5+4⍴0,K◊FRAMERECT M1◊→B15
[70]  B16:FRAMERECT M1◊OPENPICTURE B
[71]  GRAYPAT FILLRECT S5◊LC←LC+⌊PFIVE+K×(⍴WTEXT)[TWO]+S4-S2+36
[72]  OUT←(BC,RC)↑(TC,LC)↓WTEXT◊ERASERECT 1 1 ¯1 ¯1+RECT[ONE;]◊→B14
[73]  L12:→TW/B17◊B DRAWPICTURE SW◊→L2
[74]  B17:TW←ZERO◊OPTIONS[TWO;TWO]←⎕TCDC2◊OPTIONSN SETMENU OPTIONS◊→L11
     ∇
```

```
     ∇R←JOBRANK;G1;G2
[1]   R←(~IJL∊JSS,JA)/IJL◊R←(∧/CT[ONE+R;ONE+R]≥0)/R◊→(ZERO=⍴R)/ZERO◊⍎'→L',⍕HEUL
[2]   L1:G1←MPD[R]+JWT[R]◊R←R[G1⍳⌊/G1]◊→ZERO
[3]   L11:G1←MPD[R]◊→B3
[4]   L2:G1←MPD[R]×JWT[R]◊R←R[G1⍳⌈/G1]◊→ZERO
[5]   L12:G1←+\,MPD[R]×JWT[R]◊→B4
[6]   L3:G1←(,CONP[ONE;ONE+R]+CONP[ONE+R;ONE])+JWT[R]◊→(ZERO∊G1)/B6
[7]   R←R[G1⍳⌊/G1]◊→ZERO
[8]   B6:R←(G1=ZERO)/R◊G1←JWT[R]◊R←R[G1⍳⌈/G1]◊→ZERO
[9]   L13:G1←CONP[ONE;ONE+R]+CONP[ONE+R;ONE]◊→(ZERO∊G1)/B1
[10]  B3:G1←+\,JWT[R]+G1
[11]  B4:R←R[(G1≥(NONE↑G1)×(?1000)÷1000)⍳ONE]◊→ZERO
```

```
[12]   B1:R←(G1=ZERO)/R◇G1←+\,JWT[R]◇→B4
[13]   L4:G1←RE[R]×JWT[R]◇R←R[G1⍳⌈/G1]◇→ZERO
[14]   L14:G1←+\,RE[R]×JWT[R]◇→B4
[15]   L5:G1←RE2[R]×JWT[R]◇R←R[G1⍳⌈/G1]◇→ZERO
[16]   L15:G1←+\,RE2[R]×JWT[R]◇→B4
[17]   L6:G1←JWT[R]◇R←R[G1⍳⌈/G1]◇→ZERO
[18]   L16:G1←+\JWT[R]◇R←R[(G1≥(NONE↑G1)×(?1000)+1000)⍳ONE]◇→ZERO
[19]   L7:G1←T3[R]+JWT[R]◇R←R[G1⍳⌊/G1]◇→ZERO
[20]   L17:G1←T3[R]◇→B3
[21]   L8:G1←CONP[ONE+R;ONE]+JWT[R]◇→(ZERO∊G1)/B6◇R←R[G1⍳⌈/G1]◇→ZERO
[22]   L18:G1←-CONP[ONE+R;ONE]◇→(ZERO∊G1)/B1◇→B3
[23]   L9:→(ZERO≠⍴JSS)/B2◇G1←,MM[R;]
[24]   R←IZ⍴R[ONE+⌊(NONE+G1⍳⌈/G1)+JL]◇ML←(JL,JL)⍴,MM◇ML[;R]←NONE◇→ZERO
[25]   B2:G1←,ML[ST1⍳⌈/ST1;R]◇R←R[IZ⍴G1⍳⌈/G1]◇ML[;R]←NONE◇→ZERO
[26]   L19:→(ZERO≠⍴JSS)/B7◇G1←+\,⌈/MM[R;]
[27]   B15:R←R[(G1≥(NONE↑G1)×(?1000)+1000)⍳ONE]◇ML←(JL,JL)⍴,MM◇ML[;R]←NONE◇→ZERO
[28]   B7:G1←,ML[ST1⍳⌈/ST1;R]
[29]   B14:R←(G1>ZERO)/R◇G1←+\(G1>ZERO)/G1
[30]   R←R[(G1≥(NONE↑G1)×(?1000)+1000)⍳ONE]◇ML[;R]←NONE◇→ZERO
[31]   L10:G2←CONP[ONE;ONE+R]+CONP[ONE+R;ONE]◇→(ZERO∊G2)/B9
[32]   B10:→(ZERO≠⍴JSS)/B8
[33]   G1←(⌈/MM[R;])+G2◇R←R[IZ⍴G1⍳⌈/G1]◇ML←(JL,JL)⍴,MM◇ML[;R]←NONE◇→ZERO
[34]   B8:G1←(,ML[ST1⍳⌈/ST1;R])+G2◇R←R[IZ⍴G1⍳⌈/G1]◇ML[;R]←NONE◇→ZERO
[35]   B9:R←(G2=ZERO)/R◇G2←ONE◇→B10
[36]   L20:G2←CONP[ONE;ONE+R]+CONP[ONE+R;ONE]◇→(ZERO∊G2)/B13
[37]   B11:→(ZERO≠⍴JSS)/B12◇G1←+\,(⌈/MM[R;])+G2◇→B15
[38]   B12:G1←(,ML[ST1⍳⌈/ST1;R])+G2◇→B14
[39]   B13:R←(G2=ZERO)/R◇G2←ONE◇→B11
     ∇


    ⍫LINETO⍫


     ∇MAT;K;KK;G1;G2;G3;KKK;G4;G5;P;PP;G;H;N;V;W;C;A;B;F;L;MAD;MA
[1]    MA←(JL,JL)⍴ZERO◇K←ONE◇KK←TWO◇MA[ONE;ONE]←NTWO
[2]    B7:B←PD[K;]◇G1←(B<ZERO)/ICL◇→(ZERO=JT[K])/B3◇G3←⍎'C',(⍕JT[K]),⍕CL
[3]    →(ZERO=⍴G1)/B9◇G3←(~∨/G3∊G1)⌿G3
[4]    B9:N←(⍴G3)[ONE]◇G3←G3[⍋⌈/B[G3];]
[5]    B3:→(K=KK)/B12◇→(ZERO>CT[ONE+K;ONE+KK])/B12
[6]    →(NONE=×/AUDX[K,KK])/B1◇→(∨/RES<+⌿RESM[K,KK;])/B1
[7]    →(∨/ZERO=JT[K,KK])/B11◇A←PD[KK;]◇KKK←ZERO⌈JT[K]+JT[KK]-+/(A≥ZERO)∨B≥ZERO
[8]    G4←(A≥ZERO)/B◇G4←G4[⍋G4]
[9]    V←+/MPD[K,KK]◇L←V◇W←⌈/MPD[K,KK]◇→(KKK=ZERO)/B2◇→(G4[KKK]=MPD[K])/B1
[10]   B2:G2←⍎'C',(⍕JT[KK]),⍕CL◇G5←(A<ZERO)/ICL◇P←ONE◇G←G3[P;]◇F←⌈/B[G]
[11]   →(ZERO=⍴G5)/B10◇G2←(~∨/G2∊G5)⌿G2
[12]   B10:C←(⍴G2)[ONE]◇PP←C
[13]   B4:H←G2[PP;]◇V←V⌊F⌈(⌈/ZERO,B[(H∊G)/H])+⌈/A[H]◇→(V=W)/B8
[14]   PP←PP-ONE◇→(PP>ZERO)/B4◇→(P=N)/B8◇P←P+ONE◇PP←C◇G←G3[P;]◇F←⌈/B[G]◇→B4
[15]   B11:MA[K;KK]←(+/MPD[K,KK])-⌈/MPD[K,KK]◇→B13
[16]   B8:MA[K;KK]←L-V
[17]   B13:→(ZERO≤CT[ONE+KK;ONE+K])/B1
[18]   MA[K;KK]←ZERO⌈CT[ONE+KK;ONE+K]+MA[K;KK]
[19]   B1:KK←KK+ONE◇→(KK≤JL)/B3
[20]   K←K+ONE◇KK←ONE◇→(K≤JL)/B7
[21]   MAD←(+/MA>ZERO)⌊+⌿MA>ZERO◇MAD←PRI×(ONE+⌈/MAD)-MAD◇MA4←MA×MAD∘.×MAD
[22]   K←⌈/,MA4◇→(K≤ZERO)/ZERO◇MA4←((MA4≤ZERO)×MA4)+(MA4>ZERO)×MA4×99+K
[23]   MA4←MA4+ONE◇MA4←MA4+(MA4<ZERO)×NONE-MA4◇→ZERO
[24]   B12:MA[K;KK]←NTWO◇→B1
     ∇
```

```
∇MOVETO∇


   ∇MULT K;KK
[1]   JL←K⋄IJL←⍳K⋄ST←KρST⋄ST1←KρST1⋄STR←KρSTR⋄MD←100000⋄T3←TT←JLρMD
[2]   MPD←KρMPD⋄MPE←KρMPE⋄MPF←KρMPF⋄MPG←KρMPG⋄T2←TT-MPD
[3]   JS←JSS←IZ⋄PD←(K,1↓ρPD)ρPD⋄T2←KρT2⋄T3←KρT3
[4]   TT←KρTT⋄CT←(ONE+JL,JL)ρZERO,MD-MPD⋄KK←TWO
[5]   B1:CT[KK;KK]←ZERO⋄KK←KK+ONE⋄→(KK≤JL+ONE)/B1⋄CONP←CT
[6]   PUTMENUS⋄TEXTNORMAL⋄JWT←KρONE⋄PRI←KρONE⋄JT←KρJT⋄AUDX←KρAUDX
[7]   RESM←(K,FOUR)ρRESM⋄RESX←(K,FOUR)ρRESX⋄PR←(K,1↓ρPR)ρPR
   ∇


∇OPENPICTURE∇


∇PENMODE∇


∇PENNORMAL∇


∇PENPAT∇


∇PICTFRAME∇


   ∇PRIME
[1]   ⎕QLOAD WS,PRIMEDISK,PRIMEWS
   ∇


   ∇PRIORITYGET;H;C;VAL;HELPMSG;MSG2;OPT
[1]   →TW/B1⋄B DRAWPICTURE SW
[2]   B1:H←JON,'/',⍕(JL,ONE)ρPRI⋄VAL←'ENTER PRIORITY:  '
[3]   MSG2←'Priority Values'⋄HELPMSG←PRIORITYHELP
[4]   C←PRI EDA 'H VAL ',BL NA,' Priority Data'⋄→(C≡PRI)/B2
[5]   PRI←C⋄FL8←(HEAD8 CAPPEND ⍕(JL,ONE)ρPRI),AV125⋄MOVETO 176 460⋄TEXTFACE ONE
[6]   WTEXT←FL0,FL8,FL1,FL2,FL7,FL3,FL4,FL5,FL6,FL10
[7]   WTEXT←WTEXT[1 2;];'=';2 0↓WTEXT⋄WTEXT[THREE+JSS;ONE]←'*'
[8]   DRAWTEXT EIGHT↑VCEN2 +/PRI[JSS]
[9]   B2:TEXTFACE ZERO⋄→TW/ZERO⋄JOBC 11⋄B DRAWPICTURE JW
   ∇


   ∇PRS L;S1;S2;G1;G2;G3;G4;G5;G6;P;IPIX;N;S3;S4;E;F;T;KK;V;K
[1]   L←TWO↑L⋄P←MD⌊SS+360×PIX⋄G6←15.5⋄IPIX←ONE+PIX⋄→(L[ONE]<0)/B1⋄S1←16+ICL×18
[2]   S2←SEVEN+S1⋄BR←0 4ρIZ⋄M4←BR⋄S4←IZ
[3]   →((ZERO≠ρ,JS)∧(ZERO=ONE↑CS)∧L[ONE]=ONE)/B16
[4]   ERASERECT SRP[ICL,EIGHT,(~L[ONE])/EIGHT+ICL,EIGHT;]
[5]   B16:→(ZERO=ρJS)/B1⋄G1←TL<ZERO⋄G2←(G1≠ZERO,NONE↓G1)/SP1⋄→(ZERO=ρG2)/B1
[6]   G3←⌈PFIVE×ρG2⋄G2←(G3,TWO)ρG2,MD⋄G2←((G2[;ONE]<P)∧G2[;TWO]>SS)⌿G2
[7]   G3←16⌈⌊G6+IPIX×G2[;ONE]-SS⋄G4←375⌊⌊G6+IPIX×G2[;TWO]-SS⋄G1←(ρS1),ρG3
[8]   G2←(×/G1),ONE⋄→(ZERO≠CS)/B12⋄BR←161,G3,166,((ρG3),ONE)ρG4
[9]   →(TWO=+/L)/B14⋄→B1
[10]  B12:BR←(G2ρS1+ONE),(,⍉G1ρG3),(G2ρS2-ONE),,⍉G1ρG4
[11]  B1:K←ONE⋄F←SP[;CL6]⋄F←((F≥P)⍳ONE)↑F⋄G1←NTWO+(F>SS)⍳ONE
[12]  E←G1↓⍳ρF⋄F←G1↓F⋄→(L[ONE]<ZERO)/B7⋄→(CL=ZERO)/B14
[13]  B2:G1←SP[E;K]⋄G3←G1≠NONE,(NTWO↓G1),NONE⋄G2←G3/F⋄G5←NONE↓G3/G1
```

```
[14]  KK←G5≠ZERO◇G3←KK/NONE↓G2◇→(ZERO=ρG3)/B5◇G4←KK/ONE↓G2
[15]  G3←⌊G6+IPIX×ZERO⌈G3-SS◇S4←S4,KK/G5◇G4←376⌊⌊G6+IPIX×G4-SS
[16]  M4←M4⍪S1[K],G3,S2[K],((ρG4),ONE)ρG4
[17]  B5:K←K+ONE◇→(K≤CL)/B2
[18]  B14:KK←(JT[JSS]=ZERO)/JSS◇KK←((ST[KK]≤P)∧ST1[KK]>SS)/KK◇S4←S4,KK
[19]  G3←⌊G6+IPIX×ZERO⌈ST[KK]-SS◇G4←376⌊⌊G6+IPIX×ST1[KK]-SS
[20]  M4←M4⍪160,G3,167,((ρG4),ONE)ρG4
[21]  B4:→(ZERO=ρJS)/B10◇→(CS≠ZERO)/B10◇GRAYPAT FILLRECT M4◇FRAMERECT M4
[22]  BLACKPAT FILLRECT BR◇→B11
[23]  B10:BLACKPAT FILLRECT BR◇GRAYPAT FILLRECT M4◇FRAMERECT M4
[24]  B11:→L[ONE]/B7◇KK←ρS4◇→(ZERO=KK)/B7◇K←ONE◇G2←(ρM4)[ONE]
[25]  B3:MOVETO ¯2 ¯2+M4[K;1 2]◇DRAWTEXT 'J',⍕S4[K]◇K←K+ONE◇→(K≤G2)/B3
[26]  B7:→L[TWO]/ZERO◇G3←ρF◇F[G3]←F[G3]⌊P+ONE◇G3←TWO,G3◇F[ONE]←SS
[27]  V←ONE↓,⍉G3ρ⌈(F-SS)×88÷360×PIX◇G4←94◇K←ONE◇G5←297◇ERASERECT 180 16 185 375
[28]  GRAYPAT FILLRECT 203 0 204 358
[29]  B8:T←RECT[15+K;TWO]◇G1←NONE↓,⍉G3ρG5-⌈SP[E;CL+K]×G4÷RES[K]◇MOVETO G5,T
[30]  ERASERECT 204,(T+ONE),G5,T+89◇LINETO ⍉G1,[PFIVE]T+V◇K←K+ONE◇→(K≤FOUR)/B8
[31]  G1←SP[E;CL5]◇G3←G1≠NTWO,(NTWO↓G1),NTWO◇G2←G3/F◇G5←NONE↓G3/G1
[32]  KK←G5≠ZERO◇G3←KK/NONE↓G2◇→(ZERO=ρG3)/ZERO◇G4←KK/ONE↓G2
[33]  G3←⌊G6+IPIX×ZERO⌈G3-SS◇G5←KK/G5◇G4←376⌊⌊G6+IPIX×G4-SS
[34]  V←179,G3,186,((ρG4),ONE)ρG4◇G1←G5=ONE
[35]  STRIPEPAT FILLRECT G1⌿V◇GRAYPAT FILLRECT (~G1)⌿V◇FRAMERECT V
     ∇


     ∇PRTSCHED;K;KK;J;G1;G2;G3;G4;G5;B;C;D;E;F;H;UN;UN2;R;IR;M;P;OPT;P2;L
[1]   UN←172 5 284 357◇B←UN◇UN2←⎕GETBITS UN◇H←L1,L1,(SIXρL2),L3◇TEXTSIZE 12
[2]   OPENPICTURE SCREEN◇ERASERECT B◇P←1 1 1◇FRAMEROUNDRECT PRTRECTS
[3]   TEXTFACE 65◇FRAMERECT 2 4ρ172 5 284 357 174 7 282 355◇P2←PIX
[4]   R←(0 ¯2↓((ρPRTRECTS)[ONE]ρ1 0)⌿PRTRECTS)⍪256 188 275 227
[5]   MOVETO 215 281◇DRAWTEXT 'DONE'◇MOVETO 251 271◇DRAWTEXT 'CANCEL'
[6]   TEXTNORMAL◇TEXTFACE ONE◇MOVETO 203 15◇DRAWTEXT 'LISTING BY CREWMEMBER'
[7]   MOVETO 225 15◇DRAWTEXT 'LISTING BY JOB'◇DELETEMENUS
[8]   MOVETO 247 15◇DRAWTEXT 'LISTING BY TIMELINE'
[9]   MOVETO 269 15◇DRAWTEXT 'TIMELINE MINUTES/PIXEL'
[10]  MOVETO 203 178◇DRAWTEXT 'YES'◇MOVETO 203 222◇DRAWTEXT 'NO'
[11]  MOVETO 225 178◇DRAWTEXT 'YES'◇MOVETO 225 222◇DRAWTEXT 'NO'
[12]  MOVETO 247 178◇DRAWTEXT 'YES'◇MOVETO 247 222◇DRAWTEXT 'NO'
[13]  INVERTRECT R[3 5 7;]◇MOVETO 269 195◇DRAWTEXT ⍕P2
[14]  B26:FRAMERECT 2 4ρ258 190 273 225 256 188 275 227^MOVETO 269 195
[15]  TEXTFACE 1◇DRAWTEXT ⍕P2◇SCREEN DRAWPICTURE CLOSEPICTURE◇IR←⍳(ρR)[ONE]
[16]  B20:M←⎕GETKEY◇→(THREE≠ρM)/B20◇→(ONE≠M[ONE])/B20◇M←M[2 3]
[17]  L←ONE↑(M PTINRECT R)/IR◇→(ZERO=L)/B20◇→H[L]
[18]  L2:G1←⌊PFIVE×L-ONE◇→(P[G1]=TWO|L)/B20◇P[G1]←~P[G1]
[19]  INVERTRECT R[L+ZERO,NONE+TWO×P[G1];]◇→B20
[20]  L3:INVERTRECT R[L;]◇M←4 1 1 1 REPLYWINDOW 'ENTER MINUTES/PIXEL:'
[21]  →(OPT=2 3 6)/B23,B23,B24◇P2←M◇OPENPICTURE SCREEN◇ERASERECT R[L;]◇→B26
[22]  B24:HELPWINDOW PIXHELP
[23]  B23:INVERTRECT R[L;]◇→B20
[24]  L1:UN2 ⎕PUTBITS UN◇PUTMENUS◇→(L=TWO)/ZERO◇PRTSCHED2
     ∇


     ∇PRTSCHED2;S1;S2;S3;UN;UN2
[1]   UN←0 20 50 489◇UN2←⎕GETBITS UN
[2]   ⎕PRSELECT◇TEXTFACE ONE◇E←20◇F←TEN◇MOVETO E,ZERO◇S1←SP[;CL6]
[3]   DRAWTEXT 'FLIGHT PLAN ',NA◇E←E+F◇MOVETO E,0◇DRAWTEXT MH◇E←E+3×F
[4]   →(~P[1])/B21◇MOVETO E,0◇DRAWTEXT 'SCHEDULE BY CREWMEMBER'◇K←ONE◇E←E+TWO×F
[5]   B1:MOVETO E,ZERO◇DRAWTEXT 'CREWMEMBER ',(⍕K),' ',CRN[K;]◇E←E+F◇MOVETO E,0
[6]   DRAWTEXT 'TOTAL WORKLOAD = ',(⍕WL[K]),' MINUTES'◇E←E+TWO×F◇S3←S1◇S2←SP[;K]
[7]   B2:J←NONE+(S2>ZERO)⍳ONE◇S2←J↓S2◇→(ZERO=ρS2)/B3◇S3←J↓S3
[8]   MOVETO E,30◇DRAWTEXT 'JOB ',(⍕S2[ONE]),' ',JON[S2[ONE];]◇E←E+F
```

```
[9]   MOVETO E,60⋄DRAWTEXT 'START TIME = ',,CVTTD S3[ONE]⋄E←E+F
[10]  J←NONE+(S2≠S2[ONE])⍳ONE⋄S2←J↓S2⋄S3←J↓S3
[11]  MOVETO E,60⋄DRAWTEXT 'END TIME   = ',,CVTTD S3[ONE]⋄E←E+TWO×F
[12]  →(E<640)/B2⋄E←20⋄⎕←⎕TCFF⋄TEXTFACE ONE⋄→B2
[13]  B3:E←E+F⋄→(E<640)/B15⋄E←20⋄⎕←⎕TCFF⋄TEXTFACE ONE
[14]  B15:K←K+ONE⋄→(K≤CL)/B1⋄→(E=20)/B21⋄E←20⋄⎕←⎕TCFF⋄TEXTFACE ONE
[15]  B21:→(~P[TWO])/B22⋄MOVETO E,ZERO⋄DRAWTEXT 'SCHEDULE BY JOB'
[16]  E←E+TWO×F⋄K←ONE⋄G3←SP[;ICL]
[17]  B8:→(~K∊JSS)/B4⋄MOVETO E,ZERO⋄DRAWTEXT 'JOB ',(⍕K),' ',JON[K;]
[18]  E←E+F⋄G1←K=G3⋄G2←+⌿G1⋄G4←(⍴S1)|(,⍉G1)⍳ONE
[19]  MOVETO E,ZERO⋄DRAWTEXT 'START TIME = ',,CVTTD S1[G4]⋄E←E+2×F⋄J←ONE
[20]  B6:→(G2[J]=ZERO)/B5⋄MOVETO E,30
[21]  DRAWTEXT 'CREWMEMBER ',(⍕J),' ',CRN[J;],' END TIME = ',,CVTTD S1[G4+G2[J]]
[22]  E←E+F
[23]  B5:J←J+ONE⋄→(J≤CL)/B6⋄E←E+F⋄→(E<640)/B7⋄E←20⋄⎕←⎕TCFF⋄TEXTFACE ONE⋄→B7
[24]  B4:MOVETO E,0⋄DRAWTEXT 'JOB ',(⍕K),' ',JON[K;],' NOT SCHEDULED'⋄E←E+2×F
[25]  B7:K←K+ONE⋄→(K≤JL)/B8
[26]  B22:⎕←⎕TCFF⋄⎕PRUNSELECT⋄→(~P[THREE])/B28⋄C←SS⋄D←PIX⋄SS←ZERO⋄PIX←P2
[27]  B←⌈/S1×ONE,NONE↓∨/SP[;ICL]≠ZERO⋄S1←BR⋄K←CUTPICTURE SCREEN+0 0 70 70⋄S2←M4
[28]  J←14 383 31 400 0 400 33 510 199 362 209 391 168 0 178 436 0 0 11 411
[29]  J←J,207 370 218 510 25 438 33 510 32 378 168 510⋄OPENPICTURE SCREEN
[30]  ERASERECT 8 4⍴J⋄G1←CLOSEPICTURE
[31]  B10:OPENPICTURE SCREEN⋄TEXTFACE ZERO⋄MOVETO 10 0
[32]  DRAWTEXT ,8 10↑0 ¯3↓CVTTD SS+(60×PIX)×0,⍳SIX
[33]  →((ZERO=⍴JS)∧(C=SS)∧D=PIX)/B11⋄PRS ZERO
[34]  B11:G2←CLOSEPICTURE⋄OPENPICTURE 0 0 40 500⋄TEXTFACE ONE⋄MOVETO 20 20
[35]  DRAWTEXT 'FLIGHT PLAN ',NA,' PAGE ',⍕ONE+SS÷360×PIX⋄MOVETO 30 20
[36]  DRAWTEXT 'FROM ',(¯1 CVTTIME1 MS+SS),' TIME ',NONE CVTTIME1 MS+SS+360×PIX
[37]  G3←CLOSEPICTURE
[38]  E←ZERO⋄SS←SS+360×PIX⋄→(B≤SS)/B9⋄OPENPICTURE SCREEN⋄TEXTFACE ZERO
[39]  MOVETO 10 0⋄DRAWTEXT ,8 10↑0 ¯3↓CVTTD SS+(60×PIX)×ZERO,⍳SIX
[40]  →((ZERO=⍴JS)∧(C=SS)∧D=PIX)/B14⋄PRS ZERO
[41]  B14:G4←CLOSEPICTURE⋄OPENPICTURE 0 0 40 500⋄TEXTFACE ONE⋄MOVETO 20 20
[42]  DRAWTEXT 'FLIGHT PLAN ',NA,' PAGE ',⍕ONE+SS÷360×PIX⋄MOVETO 30 20
[43]  DRAWTEXT 'FROM ',(¯1 CVTTIME1 MS+SS),' TILL ',NONE CVTTIME1 MS+SS+360×PIX
[44]  G5←CLOSEPICTURE⋄E←ONE⋄SS←SS+360×PIX
[45]  B9:⎕PRSELECT
[46]  (50 50 120 120+SCREEN) DRAWPICTURE K⋄0 0 40 500 DRAWPICTURE G3
[47]  (SCREEN+50) DRAWPICTURE G1⋄(SCREEN+50) DRAWPICTURE G2
[48]  →(E=ZERO)/B12⋄(425 50 495 120+SCREEN) DRAWPICTURE K
[49]  375 0 415 500 DRAWPICTURE G5⋄(SCREEN+425 50 425 50) DRAWPICTURE G1
[50]  (SCREEN+425 50 425 50) DRAWPICTURE G4
[51]  B12:⎕←⎕TCFF⋄⎕PRUNSELECT⋄→(SS<B)/B10⋄SS←C⋄PIX←D⋄BR←S1⋄M4←S2
[52]  B28:CLEAR⋄UN2 ⎕PUTBITS UN⋄TEXTFONT FOUR⋄ERRORWINDOW 'PRINTOUT COMPLETE'
    ∇


    ∇PTINRECT∇


    ∇PUTMENUS
[1]   OPTIONSN SETMENU OPTIONS⋄PLANN SETMENU PLANMENU
[2]   CONSTN SETMENU CONSTMENU⋄MOUSEN SETMENU MOUSEMENU
[3]   DRAWMENUBAR
    ∇


    ∇R←B REPLYWINDOW A;OS;K;M;D;UN;UN2;P;KK
[1]   D←22 20 70 480⋄UN←22 20 70 484⋄UN2←⎕GETBITS UN⋄ERASERECT D⋄KK←ONE
[2]   FRAMERECT 2 4⍴D,24 22 68 478
[3]   TEXTSIZE NINE⋄TEXTFACE ONE⋄K←35+FIVE×FOUR-THREE⌊⌈(⍴,A)÷45
[4]   B1:MOVETO K,50⋄OS←45-(⌽45↑A)⍳' '⋄P←OS<ZERO⋄OS←OS+45×P⋄→((KK=3)∧OS<⍴A)/B5
```

```
[5]   DRAWTEXT DLBS2 OS↑A◊KK←KK+ONE◊A←(OS+~P)↓A◊K←K+ELEVEN◊→(ZERO≠ρA)/B1
[6]   B6:TEXTFACE ZERO
[7]   R←B GETINPUT (¯7 3+GETPEN),(ONE↑GETPEN),477◊UN2 ⎕PUTBITS UN
[8]   B4:→(ZERO≠ρ⎕GETKEY)/B4◊→ZERO
[9]   B5:DRAWTEXT (41↑A),'...'◊→B6
    ∇


    ∇RESEN R;ULC;SIZE;S1;S2;S3;S4;B;A;RECT;MS;MAX;G;M;L;IR;M1;B;MSG;G1;TR;K;H
[1]   DELETEMENUS◊G←SP[;(CL+R),CL6]◊MAX←RES[R]◊→(R=1 2 3 4)/B2,B3,B4,B5
[2]   B2:MSG←'Power Usage'◊→B6
[3]   B3:MSG←'Transmission Usage'◊→B6
[4]   B4:MSG←'Memory Usage'◊→B6
[5]   B5:MSG←'Multiplex Usage'
[6]   B6:ULC←50 50◊SIZE←200 350◊TEXTFONT ZERO◊TEXTFACE ZERO◊TEXTSIZE 12
[7]   MSG←' ',MSG,' '◊MS←TEXTWIDTH MSG◊H←L1,L2,L3,L4
[8]   B12:S1←ONE↑ULC◊S2←ONE↓ULC◊S3←S1+ONE↑SIZE◊S4←S2+ONE↓SIZE
[9]   RECT←S1,S2,S3,S4◊B←¯18 0 18 18+RECT◊UN←TWO↑B◊UN←UN,UN+16×⌈((TWO↓B)-UN)÷16
[10]  K←⎕EX 'UN2'◊UN2←⎕GETBITS UN◊OPENPICTURE B◊ERASERECT B◊TEXTSIZE 12
[11]  M1←¯14 10 ¯3 22+S1,S2,S1,S2◊TR←¯18 0 1 18+S1,S2,S1,S4
[12]  A←¯1 ¯1 18 18+S3,S4,S3,S4◊A DRAWPICTURE CORNER
[13]  K←6 4ρZERO,(TWO+S2),ZERO,S4+16◊K[;ONE]←NONE+K[;3]←S1+¯13 ¯11 ¯9 ¯7 ¯5 ¯3
[14]  RECT←RECT⍪3 4ρM1,TR,A◊FRAMERECT K⍪B⍪RECT
[15]  G1←⌊PFIVE×18+S4+S2-MS⌊S4-S2+27◊K←(S1-17),G1,(S1-ONE),S2+18+S4-G1
[16]  ERASERECT K⍪3 4ρ(0 ¯1 1 0 0 0 1 1+M1[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+M1
[17]  MOVETO (S1-THREE),G1◊TEXTFONT ZERO◊MSG DRAWMSG MS,MS⌊S4-S2+27◊TEXTNORMAL
[18]  C←(10 70+ULC),¯15 ¯20+ULC+SIZE
[19]  FRAMERECT 2 4ρ2 ¯55 15 ¯3 2 7 15 42+S3,S4,S3,S4,S3,S2,S3,S2
[20]  (SS,(SS+360×PIX),MAX,C) GRAPH2 G◊IR←⍳ONE↑ρRECT◊B DRAWPICTURE CLOSEPICTURE
[21]  B7:M←⎕GETKEY◊→(THREE≠ρM)/B7◊→(TWO=M[ONE])/B7◊M←M[2 3]
[22]  L←ONE↑(M PTINRECT RECT)/IR◊→(ZERO=L)/B7◊→H[L]
[23]  L1:→(~BUTTON)/B7
[24]  E←SS+⌊PFIVE+360×PIX×((C[2]⌈C[4]⌊GETMOUSE[TWO])-C[2])÷SIZE[TWO]-90
[25]  MOVETO 12 10+S3,S2◊DRAWTEXT NFIVE↑,CVTTD E◊MOVETO 12 ¯52+S3,S4
[26]  DRAWTEXT EIGHT↑VCEN2 G[NONE+(G[;TWO]≤E)⍳ZERO;ONE]◊→L1
[27]  L2:→BUTTON/L2◊→(~GETMOUSE PTINRECT RECT[L;])/B7◊UN2 ⎕PUTBITS UN◊PUTMENUS
[28]  →ZERO
[29]  L3:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
[30]  B10:→(~BUTTON)/B11◊FRAMERECT M1◊M1←B+4ρ(0 0⌈GETMOUSE)-M◊FRAMERECT M1◊→B10
[31]  B11:ULC←ULC+((0 0⌈GETMOUSE)-M)
[32]  B15:FRAMERECT M1◊ERASERECT B◊PENPAT BLACKPAT◊PENMODE EIGHT
[33]  UN2 ⎕PUTBITS UN◊→B12
[34]  L4:M1←B◊PENPAT GRAYPAT◊PENMODE TEN
[35]  B13:→(~BUTTON)/B14◊FRAMERECT M1
[36]  M1←(TWO↑B),(18 18+ULC)+80 105⌈SIZE+GETMOUSE-M◊FRAMERECT M1◊→B13
[37]  B14:SIZE←80 105⌈SIZE+GETMOUSE-M◊→B15
    ∇


    ∇RESENL R;UN;UN2;H;C;E
[1]   RESEN R
    ∇


    ∇B RESPRT A;K;G1
[1]   ⎕←⎕TCNL,B◊K←TWO◊→(ZERO=PCTR[ONE])/B1
[2]   CUR 36◊'Crew = ',2↓2⍕A[ONE]
[3]   B1:→(ZERO=PCTR[K])/B2◊CUR 30◊'Resource ',(⍕K-ONE),' = ',2↓2⍕A[K]
[4]   B2:K←K+ONE◊→(K≤FIVE)/B1◊→(NRES≤ONE)/ZERO
[5]   CUR 33◊'AVERAGE = ',2↓2⍕G1←(+/(PCTR>ZERO)/A)÷NRES
[6]   CUR 32◊'VARIANCE = ',2↓4⍕(+/(PCTR>ZERO)/(A-G1)×A-G1)÷NRES-ONE
```

```
∇

∇B RESPRT2 A;K;G1
[1]  ⎕←⎕TCNL,B◊K←TWO◊→(ZERO=PCTR[ONE])/B1
[2]  CUR 36◊'Crew = ',2↓4⍕A[ONE]
[3]  B1:→(ZERO=PCTR[K])/B2◊CUR 30◊'Resource ',(⍕K-ONE),' = ',2↓4⍕A[K]
[4]  B2:K←K+ONE◊→(K≤FIVE)/B1◊→(NRES≤ONE)/ZERO
[5]  CUR 35◊'TOTAL = ',2↓4⍕+/(PCTR>ZERO)/A
∇


∇S;FL0;FL1;FL2;FL3;FL4;FL5;FL6;FL7;CL;ICL;PD;CONP;IR;JS;SR;MH;AUTV
[1]  ⎕SA←ACTIONSTOP◊T◊INIT1 MAN
∇


⍒SCROLLRECT⍒


∇R←C SELWINDOW E;M;D;UN;UN2;G1;G2;K
[1]  DELETEMENUS
[2]  B3:UN←222 20 270 340◊G1←⍳ONE↑⍴C◊UN2←⎕GETBITS UN
[3]  ERASERECT UN◊FRAMERECT 2 4⍴UN,224 22 268 338◊K←,E[ONE]◊E←E[TWO]
[4]  MOVETO ¯2 ¯2+TWO↑,C[K;]◊DRAWTEXT 'J',⍕JS◊C[K;]←ZERO
[5]  R←(ONE↑⍴C)⍴ZERO◊R[K]←ONE◊K←ONE
[6]  B2:MOVETO 250 90◊ERASERECT 225 23 267 337
[7]  TEXTFACE ONE◊DRAWTEXT 'SELECT ',(⍕E-K),' MORE CREWMEMBER',(ONE≠E-K)↑'S'
[8]  B1:M←⎕GETKEY◊→(THREE≠⍴M)/B1◊→(TWO=M[ONE])/B1◊M←M[2 3]
[9]  G2←ONE↑(M PTINRECT C)/G1◊→(ZERO=G2)/B1◊R[G2]←ONE◊GRAYPAT FILLRECT C[G2;]
[10] MOVETO ¯2 ¯2+TWO↑,C[G2;]◊TEXTFACE ZERO◊DRAWTEXT 'J',⍕JS◊FRAMERECT C[G2;]
[11] C[G2;]←ZERO◊K←K+ONE◊→(E≠K)/B2◊C[;ONE]←C[;ONE]+ONE◊C[;FOUR]←C[;FOUR]⌊375
[12] C[;THREE]←C[;THREE]-ONE◊C[;TWO]←C[;TWO]⌈16◊ERASERECT C◊UN2 ⎕PUTBITS UN
[13] B5:→(ZERO≠⍴⎕GETKEY)/B5◊PUTMENUS
∇


∇R←C SELWINDOW2 E;M;UN;UN2;G1;G2;K
[1]  DELETEMENUS◊G1←⍳EIGHT
[2]  B3:UN←222 20 270 340◊UN2←⎕GETBITS UN◊ERASERECT UN
[3]  FRAMERECT 2 4⍴UN,224 22 268 338◊R←,C◊TEXTFACE ONE
[4]  B2:MOVETO 250 90◊ERASERECT 225 23 267 337
[5]  DRAWTEXT 'SELECT ',(⍕E-⍴R),' MORE CREWMEMBER',(ONE≠E-⍴R)↑'S'
[6]  B1:M←⎕GETKEY◊→(THREE≠⍴M)/B1◊→(TWO=M[ONE])/B1◊M←M[2 3]
[7]  G2←ONE↑(M PTINRECT SRP2)/G1◊→(ZERO=G2)/B1◊→(G2>CL)/B1◊→(PDS[G2]<ZERO)/B6
[8]  →(G2∊R)/B6◊R←R,G2◊→(E≠⍴R)/B2◊UN2 ⎕PUTBITS UN
[9]  B5:→(ZERO≠⍴⎕GETKEY)/B5◊PUTMENUS◊TEXTFACE ZERO◊→ZERO
[10] B6:⎕SOUND BEEP◊→B1
∇


∇SETCON;WT;SIZE;ULC;TC;LC;RC;BC;MS;MSG;D;S1;S2;S3;S4;G1;G2;G3;G4;G5;CR;HS
[1]  IR←⍳TEN◊L←ZERO◊F←IZ◊A←L1,L2,L3,L4,L5,L6,L7,L8,L9,L10◊K←(TWO+JL),SIX
[2]  TC4←(K↑HE4),AV125◊TC5←(K↑HE5),AV125◊TC6←(K↑HE6),AV125◊TC7←(K↑HE7),AV125
[3]  T1←-CONP[ONE+IJL;ONE]◊TC8←K↑HE8
[4]  TC1←(HE1 CAPPEND CVTTD T1),AV125◊TC2←(HE2 CAPPEND CVTTD T2),AV125
[5]  TC3←(HE3 CAPPEND CVTTD T3),AV125◊DELETEMENUS◊TEXTSIZE 12◊TEXTFONT 0
[6]  WT←TC1,TC2,TC3,TC4,TC5,TC6,TC7,TC8◊TEXTFACE ZERO◊WT[TWO;]←'='◊LC←ZERO
[7]  TC←LC◊SIZE←75 575⌊10 6×1 15+⍴WT◊ULC←100 0◊WT1←(ONE↓⍴WT)+ONE↓⍴TC0
```

```
MSG←' ',NA,' Time Constraints '⋄MS←TEXTWIDTH MSG
B4:S1←ULC[ONE]⋄S2←ULC[TWO]⋄S3←S1+SIZE[ONE]⋄D←¯9 7 1 7+S1,S2,S1,S2
S4←S2+SIZE[TWO]⋄BC←⌊(NTHREE+SIZE[ONE])+11⋄RC←⌊(NNINE+SIZE[TWO])+SIX
RECT←S1,S2,S3,S4⋄ER←1 1 ¯1 ¯1+RECT⋄B←¯18 0 18 18+RECT⋄UN←TWO↑B⋄K←⎕EX 'UN2'
UN←UN,UN+16×⌈((TWO↓B)-UN)+16⋄UN2←⎕GETBITS UN⋄TR←¯18 0 1 18+S1,S2,S1,S4
ERASERECT B⋄CR←¯14 10 ¯3 22+S1,S2,S1,S2⋄G1←0 ¯1 18 18+S1,S4,S1,S4
G1 DRAWPICTURE UPARROW⋄G2←¯18 ¯1 0 18+S3,S4,S3,S4⋄G2 DRAWPICTURE DOWNARROW
G3←¯1 0 18 18+S3,S2,S3,S2⋄G3 DRAWPICTURE LEFTARROW
G4←¯1 ¯18 18 0+S3,S4,S3,S4⋄G4 DRAWPICTURE RIGHTARROW
G5←¯1 ¯1 18 18+S3,S4,S3,S4⋄G5 DRAWPICTURE CORNER
K←(S1-15)+TWO×⍳SIX⋄K←⍉4 6ρ(K-ONE),(SIXρTWO+S2),K,SIXρS4+16
RECT←RECT⍪9 4ρCR,TR,G1,G2,G3,G4,G5⋄FRAMERECT K⍪B⍪¯1 0↓RECT
G1←⌈PFIVE×15+S4+S2-MS⌊S4-S2+30⋄K←(S1-17),G1,(S1-ONE),S2+18+S4-G1
ERASERECT K⍪3 4ρ(0 ¯1 1 0 0 0 1 1+CR[1 2 3 2 1 4 3 4]),1 1 ¯1 ¯1+CR
MOVETO (S1-THREE),G1⋄TEXTSIZE 12⋄TEXTFONT ZERO⋄MSG DRAWMSG MS,MS⌊S4-S2+30
TEXTNORMAL⋄S5←0 18 17 ¯18+S3,S2,S3,S4⋄S6←18 0 ¯18 17+S1,S4,S3,S4
B14:MOVETO 10 8+S1,S2⋄DRAWTEXT (BC,RC)↑(TC,ZERO)↓TC0,(ZERO,LC)↓WT
→(ZERO=ρF)/B33⋄→((F≤TC-TWO)∨F>TC+BC-TWO)/B33⋄K←11×F+TWO-TC
INVERTRECT D+K,ZERO,K,ONE+SIX×RC⌊NONE+(ρTC0)[TWO]
B33:→(∨/L=1 6 7 10)/B18⋄GRAYPAT FILLRECT S6
G1←(BC×¯36+S3-S1)+TWO×JL+TWO⋄G2←S1+18+((TC+PFIVE×BC)+TWO+JL)×(S3-S1)-36
S6←(⌊G2-G1),S4,((S3-18)⌊⌊G2+G1),17+S4⋄ERASERECT S6⋄FRAMERECT S6
RECT[NINE;]←S6⋄→(∨/L=4 5 9)/B2
B18:GRAYPAT FILLRECT S5
G5←(RC×¯36+S4-S2)+TWO×WT1⋄G2←S2+18+((LC+PFIVE×RC)+WT1)×(S4-S2)-36
S5←S3,(⌊G2-G5),(S3+17),(S4-18)⌊⌊G2+G5⋄ERASERECT S5⋄FRAMERECT S5
RECT[TEN;]←S5
B2:M←⎕GETKEY⋄→(THREE≠ρM)/B2⋄→(TWO=M[ONE])/B2⋄M←M[2 3]
L←ONE↑(M PTINRECT RECT)/IR⋄→(ZERO=L)/B2⋄→A[L]
B3:⎕SOUND BEEP⋄→B2
L1:→BUTTON/L1⋄→(M PTINRECT S1,S2,S3,S2+FOUR+SIX×(ρTC0)[TWO])/B6⋄IRC
ERASERECT ER⋄→B14
B6:K←ONE↑F⋄IRCON⋄→(K=ONE↑F)/B2⋄ERASERECT ER⋄→B14
L2:→BUTTON/L2⋄UN2 ⎕PUTBITS UN⋄PUTMENUS
RE2←+/(⍉(1 0↓0 1↓CT)≤ZERO)×(JL,JL)ρRE⋄→ZERO
L3:M1←B⋄PENPAT GRAYPAT⋄PENMODE TEN
B7:→(~BUTTON)/B5⋄FRAMERECT M1⋄M1←B+4ρ(0 0⌈GETMOUSE)-M⋄FRAMERECT M1⋄→B7
B5:ULC←ULC+((0 0⌈GETMOUSE)-M)
B30:FRAMERECT M1⋄PENPAT BLACKPAT⋄UN2 ⎕PUTBITS UN⋄PENMODE EIGHT⋄→B4
L4:→(TC≤ZERO)/B3⋄TC←TC-ONE⋄ERASERECT ER⋄→B14
L5:→(TC≥JL-BC-TWO)/B3⋄TC←TC+ONE⋄ERASERECT ER⋄→B14
L6:→(LC≤ZERO)/B3⋄HS←(⌽(LC-ONE)↑WT[ONE;])⍳AV125⋄G3←LC⌊HS⋄LC←LC-G3
ERASERECT ER⋄→B14
L7:G3←WT1-RC⋄→(LC≥G3)/B3⋄HS←((RC+LC+ONE-ONE↓ρFL0)↓WT[ONE;])⍳AV125
LC←LC+HS⌊G3⋄ERASERECT ER⋄→B14
L8:M1←B⋄PENPAT GRAYPAT⋄PENMODE 10⋄K←ZERO
B8:→(~BUTTON)/B9⋄FRAMERECT M1⋄M1←B[1 2],(18 18+ULC)+36 36⌈⌈SIZE+GETMOUSE-M
FRAMERECT M1⋄→B8
B9:SIZE←36 36⌈SIZE+GETMOUSE-M⋄→B30
L9:M1←S6⋄PENPAT GRAYPAT⋄PENMODE TEN⋄K←ZERO
B21:→(~BUTTON)/B22⋄FRAMERECT M1
K←(18+S1-S6[1])⌈(S3-18+S6[3])⌊1↑GETMOUSE-M⋄M1←S6+4ρK,0⋄FRAMERECT M1⋄→B21
B22:FRAMERECT M1⋄TC←TC+⌊PFIVE+K×(ρWT)[ONE]+S3-S1+36
B31:ERASERECT ER⋄PENPAT BLACKPAT⋄PENMODE EIGHT⋄→B14
L10:M1←S5⋄PENPAT GRAYPAT⋄PENMODE TEN⋄K←ZERO
B23:→(~BUTTON)/B24⋄FRAMERECT M1
K←(18+S2-S5[2])⌈(S4-18+S5[4])⌊1↓GETMOUSE-M⋄M1←S5+4ρ0,K⋄FRAMERECT M1⋄→B23
B24:FRAMERECT M1⋄LC←LC+⌊PFIVE+K×WT1+S4-S2+36⋄→B31
∇


∇SETCON1;TR;B;RECT;K;KK;WT1;S5;S6;A;L;M;IR;M1;ER;F;W4;W5;W6;W7;TC4;TC5
SETCON2
```

```
    ∇

    ∇SETCON2;TC6;TC7;TC1;TC2;TC3;UN;UN2;T1;TC8
[1]  SETCON
    ∇


    ∇SETJOB R;E;F;H;C;AUDG;RESG;G1;G2;K
[1]  AUDG←⍎R,'JOBAUDIO'⋄AUDX←AUDG-THREE⋄AUDX[(AUDX=NTWO)/IJL]←ZERO
[2]  RESG←⍎R,'JOBRES'⋄RESM←ZERO⌈RESG⋄E←+⌿RESM
[3]  RES←RES⌊E⋄F←(RES<ZERO)/1 2 3 4⋄RES[F]←E[F]
[4]  RECT[16 17 18 19;]←⍉4 4⍴(FOUR⍴188),0 90 180 270,(FOUR⍴298),90 180 270 360
[5]  F←RES≠E⋄FL0←(HEAD0 CAPPEND ' ',JON),AV125⋄H←JL,ONE⋄E←⍕H⍴JT
[6]  E[(JT=JOBSIZEDEF)/IJL;]←' '⋄RESX←((JL,FOUR)⍴RES)-RESM
[7]  FL1←(HEAD1 CAPPEND E),AV125⋄F←⍎R,'JOBTDEF'⋄MPD←⍎R,'MPD'
[8]  K←RESM×⍉(FOUR,JL)⍴MPD
[9]  RE←(JT×MPD×(+/JT×MPD)+CL×CL)++/K×(JL,FOUR)⍴(+⌿K)+RES×RES⋄RE2←RE
[10] E←⍕H⍴F⋄E[(F=JOBTIMEDEF)/IJL;]←' '⋄FL2←(HEAD2 CAPPEND E),AV125
[11] PD←⍎R,'PD'⋄PRC←IJL⋄NEC←IJL⋄TAR←-IJL⋄CONP←(ONE+JL,JL)⍴ZERO,MD-MPD⋄MPE←⌈/PD
[12] CONP←CONP-CONP×(IJL,JL+ONE)∘.=IJL,JL+ONE⋄E←⍕H⍴MPD⋄CT←CONP
[13] E[(MPD=JOBTIMEDEF)/IJL;]←' '⋄FL7←(HEAD7 CAPPEND E),AV125⋄F←RESG[;,ONE]
[14] E←⍕F⋄E[(,F=JOBPOWERDEF)/IJL;]←' '⋄FL3←(HEAD3 CAPPEND E),AV125
[15] F←RESG[;,TWO]⋄G1←RESG[;,THREE]⋄G2←RESG[;,FOUR]
[16] E←⍕F⋄E[(,F=JOBTRANSDEF)/IJL;]←' '⋄FL4←(HEAD4 CAPPEND E),AV125
[17] E←⍕G1⋄E[(,G1=JOBMEMORYDEF)/IJL;]←' '⋄FL5←(HEAD5 CAPPEND E),AV125
[18] E←⍕G2⋄E[(,G2=JOBMULTIPLEXDEF)/IJL;]←' '⋄T3←JL⍴MD⋄TT←T3
[19] FL6←(HEAD6 CAPPEND E),AV125⋄E←AUDION[AUDG;]⋄FL10←HEAD10 CAPPEND E
[20] FL8←(HEAD8 CAPPEND ⍕(JL,ONE)⍴PRI),AV125⋄RES←RES+RES=ZERO
[21] TC0←(HE0 CAPPEND JON),AV125⋄T2←MD-MPD⋄OUT←IZ⋄D←IZ⋄MPF←MPD⋄MPG←MPE
[22] TC0[TWO;]←'='⋄SW←IZ⋄JW←IZ⋄RC←IZ⋄BC←IZ⋄S1←IZ⋄S2←IZ⋄S3←IZ⋄S4←IZ⋄JT←ZERO⌈JT
    ∇


  ∇SETMENU∇


    ∇R←SETPIX;RECT;G1;G2;G3;SSOLD;PIXOLD;IR;L;M;B;G4;OPT;UN;UN2;H
[1]  DELETEMENUS
[2]  R←ZERO⋄B←192 5 290 350⋄UN←192 5 304 357⋄UN2←⎕GETBITS UN⋄H←L1,L2,L3,L4,L5
[3]  OPENPICTURE SCREEN⋄ERASERECT B⋄FRAMERECT 2 4⍴192 5 290 350 194 7 288 348
[4]  G1←213 260 233 340 15 15,SPRECTS[ONE;],17 17,249 260 269 340 15 15
[5]  FRAMEROUNDRECT 4 6⍴G1,SPRECTS[TWO;],17 17⋄TEXTSIZE 12⋄TEXTFACE 65
[6]  MOVETO 228 281⋄DRAWTEXT 'DONE'⋄MOVETO 264 271⋄DRAWTEXT 'CANCEL'
[7]  TEXTSIZE NINE⋄SSOLD←SS⋄PIXOLD←PIX⋄IR←⍳ONE↑⍴SPRECTS
[8]  MOVETO 206 70⋄DRAWTEXT 'START DATE = ',NONE CVTTIME1 MS
[9]  MOVETO 216 70⋄DRAWTEXT 'DURATION   =',,CVTTD MD
[10] B2:TEXTFACE ONE
[11] G1←SS⌈MD-360×PIX⋄G4←,0 1440⊤G1⋄G2←⌈(MD-SS)÷360⋄G1←60×⌈G1÷60⋄G3←,0 1440⊤SS
[12] MOVETO 235 10⋄DRAWTEXT 'MINUTES/PIXEL = ',(⍕PIX)
[13] MOVETO 236 167⋄DRAWTEXT '(MAX = ',(⍕G2),')'
[14] MOVETO 255 10⋄DRAWTEXT 'DISPLAY START DAY = ',(⍕G3[ONE])
[15] MOVETO 268 10⋄DRAWTEXT '                  HOUR = ',(⍕G3[TWO]÷60)
[16] MOVETO 281 10⋄DRAWTEXT '                  (MAX = ',(NTHREE↓,CVTTD G1),')'
[17] FRAMERECT SPRECTS[3 4 5;]⋄TEXTNORMAL⋄SCREEN DRAWPICTURE CLOSEPICTURE
[18] B1:M←⎕GETKEY⋄→(3≠⍴M)/B1⋄→(ONE≠M[ONE])/B1⋄M←M[2 3]
[19] L←ONE↑(M PTINRECT SPRECTS)/IR⋄→(ZERO=L)/B1⋄→H[L]
[20] L1:UN2 ⎕PUTBITS UN⋄PUTMENUS⋄→ZERO
[21] L2:SS←SSOLD⋄PIX←PIXOLD⋄R←ONE⋄→L1
[22] L3:INVERTRECT SPRECTS[L;]
[23] M←(4 1 1 1,G2) REPLYWINDOW 'ENTER MINUTES/PIXEL:'⋄→(OPT=2 3 6)/B3,B3,B5
```

```
[24]   PIX←M◊OPENPICTURE SCREEN◊ERASERECT 225 10 282 250◊→B2
[25]   L4:INVERTRECT SPRECTS[L;]◊M←(ONE↑G4)-(G3[TWO]+60)>⌈G4[TWO]+60
[26]   M←(4 1 1 0,M) REPLYWINDOW 'ENTER WINDOW START (DAYS):'
[27]   →(OPT=2 3 6)/B3,B3,B6◊SS←(M×1440)+G3[TWO]
[28]   B4:OPENPICTURE SCREEN◊ERASERECT 225 10 282 250◊→B2
[29]   L5:INVERTRECT SPRECTS[L;]◊M←G3[ONE]=G4[ONE]◊M←(23×~M)+M×⌈G4[TWO]+60
[30]   M←(4 1 1 0,M) REPLYWINDOW 'ENTER WINDOW START (HOURS):'
[31]   →(OPT=2 3 6)/B3,B3,B7◊SS←(1440×G3[ONE])+60×M◊→B4
[32]   B5:HELPWINDOW PIXHELP◊→B3
[33]   B6:HELPWINDOW SSHELP1◊→B3
[34]   B7:HELPWINDOW SSHELP2
[35]   B3:INVERTRECT SPRECTS[L;]◊→B1
     ∇


     ∇SETSCHED;VAL;K;G1;G2
[1]    CS←JT[JS]◊PDS←(CL⌈EIGHT×CS=ZERO)⍴PD[JS;]◊TEXTFACE ONE◊MOVETO 216 370
[2]    →(ZERO=CS)/B1
[3]    G2←23+18×⍳CL◊SR2←⍉(FOUR,CL)⍴(16+18×⍳CL),(CL⍴400),G2,400+ZERO⌈⌈PDS÷PIX
[4]    B2:DRAWTEXT (⍕JS),' ',JON[IZ⍴JS;]◊MOVETO 33 443◊DRAWTEXT 'CREW ='
[5]    MOVETO 32 490◊DRAWTEXT ⍕CS◊DRAWLINE 23 490 23 496
[6]    DRAWLINE 24 443 24 453◊CONPROC◊→ACS/B8
[7]    B10:PRS 1 1◊K←ONE◊TEXTFACE ONE
[8]    B4:→(CS=ZERO)/B3◊G1←PDS[K]◊→(ZERO>G1)/B6◊MOVETO G2[K],378◊DRAWTEXT ⍕G1
[9]    MOVETO G2[K],480◊DRAWTEXT ⍕WL[K]
[10]   B6:K←K+ONE◊→(CL≥K)/B4◊RECT[19+⍳CL;]←SR2
[11]   B5:TEXTFACE ZERO◊FRAMERECT SR2◊SCREEN DRAWPICTURE CLOSEPICTURE◊→ZERO
[12]   B8:SCREEN DRAWPICTURE CLOSEPICTURE◊CSCHED◊→(ZERO=⍴JS)/ZERO
[13]   →(AJS∧ACS)/ZERO◊OPENPICTURE SCREEN◊→B10
[14]   B1:SR2←8 4⍴(28⍴ZERO),160 400 167,400+ZERO⌈⌈(ONE↑PDS)÷PIX◊→B2
[15]   B3:MOVETO 167 378◊DRAWTEXT ⍕ONE↑PDS◊RECT[19+⍳EIGHT;]←SR2◊
[16]   MOVETO 167 480◊K←(JT[JSS]=ZERO)/JSS◊DRAWTEXT ⍕+/ST1[K]-ST[K]◊→B5
     ∇


     ∇R←SETSEARCH;B;UN;UN2;H;G1;IR;M;L;OCOLD;OPT
[1]    DELETEMENUS◊R←ZERO◊B←96 5 290 446◊UN←96 5 304 453◊UN2←⎕GETBITS UN
[2]    H←L1,L2,L3,L3,L4,L5,L6,L7,L8,L8
[3]    OPENPICTURE B◊ERASERECT B◊FRAMERECT 2 4⍴96 5 290 446 98 7 288 444
[4]    G1←173 352 193 432 15 15,SRRECTS[ONE;],17 17,209 352 229 432 15 15
[5]    FRAMEROUNDRECT 4 6⍴G1,SRRECTS[TWO;],17 17◊TEXTSIZE 12◊TEXTFACE 65
[6]    MOVETO 188 377◊DRAWTEXT 'DONE'◊MOVETO 224 367◊DRAWTEXT 'CANCEL'◊TEXTNORMAL
[7]    IR←⍳ONE↑⍴SRRECTS◊OCOLD←OC◊MOVETO 124 20◊DRAWTEXT 'SEARCH TYPE:'
[8]    MOVETO 166 20◊DRAWTEXT 'SEARCH METHOD:'
[9]    MOVETO 208 20◊DRAWTEXT 'SEARCH TIME:'
[10]   MOVETO 249 20◊DRAWTEXT 'OPTIMALITY CRITERIA:'
[11]   TEXTFACE ZERO◊MOVETO 138 40◊DRAWTEXT 'DETERMINISTIC'
[12]   MOVETO 152 40◊DRAWTEXT 'RANDOMIZED'
[13]   MOVETO 180 40◊DRAWTEXT 'NUMBER OF TRIALS     ='
[14]   MOVETO 194 40◊DRAWTEXT 'ITERATIONS PER TRIAL ='
[15]   MOVETO 221 40◊DRAWTEXT 'MAX SEARCH TIME FROM START ='
[16]   MOVETO 235 40◊DRAWTEXT 'MAX SEARCH TIME FROM BEST  ='◊MOVETO 264 40
[17]   DRAWTEXT 'MINIMIZE COMPLETION TIME OF LAST JOB'◊MOVETO 277 40
[18]   DRAWTEXT 'MINIMIZE AVERAGE COMPLETION TIME'◊FRAMERECT 2 0↓SRRECTS
[19]   B2:→(OC[ONE]<ZERO)/B16
[20]   MOVETO 221,TWO+SRRECTS[SEVEN;TWO]◊DRAWTEXT NNINE↑,CVTID OC[ONE]
[21]   B16:→(OC[TWO]<ZERO)/B17◊MOVETO 235,TWO+SRRECTS[EIGHT;TWO]
[22]   DRAWTEXT NNINE↑,CVTID OC[TWO]
[23]   B17:G1←SRRECTS[EIGHT+OC[THREE];1 2]◊MOVETO G1+6 5◊LINETO G1+10 9
[24]   LINETO G1+2 13◊G1←SRRECTS[TWO+OC[SIX];1 2]◊MOVETO G1+6 5◊LINETO G1+10 9
[25]   LINETO G1+2 13◊MOVETO 180 185◊DRAWTEXT NNINE↑⍕OC[FOUR]◊MOVETO 194 185
[26]   DRAWTEXT NNINE↑⍕OC[FIVE]◊B DRAWPICTURE CLOSEPICTURE
```

```
[27]  B1:M←⎕GETKEY⋄→(3≠ρM)/B1⋄→(ONE≠M[ONE])/B1⋄M←M[2 3]
[28]  L←ONE↑(M PTINRECT SRRECTS)/IR⋄→(ZERO=L)/B1⋄→H[L]
[29]  L1:UN2 ⎕PUTBITS UN⋄PUTMENUS⋄→ZERO
[30]  L2:OC←OCOLD⋄R←ONE⋄→L1
[31]  L3:→(OC[SIX]=L-TWO)/B1⋄OPENPICTURE B
[32]  ERASERECT 1 1 ¯1 ¯1+,SRRECTS[OC[SIX]+TWO;]⋄OC[SIX]←L-TWO⋄→B2
[33]  L4:INVERTRECT 1 1 ¯1 ¯1+,SRRECTS[L;]
[34]  M←4 1 1 1 REPLYWINDOW 'ENTER NUMBER OF TRIALS:'⋄→(OPT=2 3 6)/B3,B21,B22
[35]  OC[L-ONE]←M⋄→B18
[36]  L5:INVERTRECT 1 1 ¯1 ¯1+,SRRECTS[L;]
[37]  M←4 1 1 1 REPLYWINDOW 'ENTER NUMBER OF ITERATIONS PER TRIAL:'
[38]  →(OPT=2 3 6)/B3,B21,B23⋄OC[L-ONE]←M⋄→B18
[39]  B21:OC[L-ONE]←ONE⋄→B18
[40]  B22:HELPWINDOW SEARCHHELP3⋄→B3
[41]  HELPWINDOW SEARCHHELP4⋄→B3
[42]  L6:INVERTRECT 1 1 ¯1 ¯1+,SRRECTS[L;]
[43]  M←8 0 0 REPLYWINDOW 'ENTER SEARCH TIME FROM START:'
[44]  →(OPT=2 3 6)/B3,B4,B7⋄OC[L-SIX]←M
[45]  B18:OPENPICTURE B⋄ERASERECT 1 1 ¯1 ¯1+,SRRECTS[L;]⋄→B2
[46]  L7:INVERTRECT 1 1 ¯1 ¯1+,SRRECTS[L;]
[47]  M←8 0 0 REPLYWINDOW 'ENTER SEARCH TIME FROM BEST:'
[48]  →(OPT=2 3 6)/B3,B4,B8⋄OC[L-SIX]←M⋄→B18
[49]  B4:OC[L-SIX]←NONE⋄→B18
[50]  B7:HELPWINDOW SEARCHHELP1⋄→B3
[51]  B8:HELPWINDOW SEARCHHELP2
[52]  B3:INVERTRECT 1 1 ¯1 ¯1+,SRRECTS[L;]⋄→B1
[53]  L8:→(OC[THREE]=L-EIGHT)/B1⋄OPENPICTURE B
[54]  ERASERECT 1 1 ¯1 ¯1+,SRRECTS[OC[THREE]+EIGHT;]⋄OC[THREE]←L-EIGHT⋄→B2
    ∇
```

```
    ∇R←SETTAR J;RECT;G1;G2;G3;K;IR;L;M;B;G4;OPT;UN;UN2;TR;BC;DEF;H
[1]   R←ZERO⋄B←192 5 290 421⋄OPENPICTURE B⋄UN←192 5 304 421⋄UN2←⎕GETBITS UN
[2]   BC←ZERO⋄RECT←(2 4ρ223 308 247 392 254 308 278 392)⍪4 0↓TARRECTS
[3]   H←L1,L2,L3,L4,L5,L6,L3,L4,L5,L10⋄G2←RECT[4 5 8 9 3 7;]+6 4ρ1 1 ¯1 ¯1
[4]   ERASERECT B⋄FRAMERECT 2 4ρ192 5 290 421 194 7 288 419
[5]   G1←225 310 245 390 15 15 223 308 247 392 17 17 256 310 276 390 15 15
[6]   FRAMEROUNDRECT 4 6ρG1,254 308 278 392 17 17⋄TEXTSIZE 12⋄TEXTFACE 65
[7]   MOVETO 240 331⋄DRAWTEXT 'DONE'⋄MOVETO 271 321⋄DRAWTEXT 'CANCEL'⋄IR←ιTEN
[8]   TEXTSIZE NINE⋄MOVETO 205 20⋄DRAWTEXT 'JOB = ',JON[J;]
[9]   MOVETO 216 20⋄K←'EST =',(,CVTTD TCEST J),'   LST =',,CVTTD TCLST J
[10]  DRAWTEXT K,'   LET =',,CVTTD T3[J]⋄MOVETO 226 42
[11]  DRAWTEXT 'ENTER INFEASIBLE TIME WINDOWS'
[12]  MOVETO 238 30⋄DRAWTEXT 'DEL WIN-START   WIN-END   SCROLL'
[13]  242 252 260 271 DRAWPICTURE UPARROW⋄260 252 278 271 DRAWPICTURE DOWNARROW
[14]  FRAMERECT TARRECTS⋄TEXTFACE ZERO
[15]  TR←(TARι-J)↓(NONE+TARι-J+ONE)↑TAR⋄TR←((PFIVE×ρTR),TWO)ρTR
[16]  B5:→(BC>NONE+(ρTR)[ONE])/B13⋄MOVETO 254 35⋄DRAWTEXT ⍕BC+ONE
[17]  MOVETO 254 65⋄DRAWTEXT ,CVTTD TR[BC+ONE;ONE]⋄MOVETO 254 151
[18]  DRAWTEXT ,CVTTD TR[BC+ONE;TWO]⋄→(BC>NTWO+(ρTR)[ONE])/B13⋄MOVETO 272 35
[19]  DRAWTEXT ⍕BC+TWO⋄MOVETO 272 65⋄DRAWTEXT ,CVTTD TR[BC+TWO;ONE]
[20]  MOVETO 272 151⋄DRAWTEXT ,CVTTD TR[BC+TWO;TWO]
[21]  B13:B DRAWPICTURE CLOSEPICTURE
[22]  B1:M←⎕GETKEY⋄→(3≠ρM)/B1⋄→(ONE≠M[ONE])/B1⋄M←M[2 3]
[23]  L←ONE↑(M PTINRECT RECT)/IR⋄→(ZERO=L)/B1⋄→H[L]
[24]  B2:⎕SOUND BEEP⋄→B1
[25]  L1:TR←TR[⍋TR[;ONE];]⋄TAR←((TARι-J)↑TAR),(,TR),(NONE+TARι-J+ONE)↓TAR
[26]  B14:UN2 ⎕PUTBITS UN⋄→ZERO
[27]  L2:R←ONE⋄→B14
[28]  L3:L←.25×L+ONE⋄→((ρTR)[ONE]<BC+L)/B2⋄TR←TR ELIM BC+L
[29]  B12:OPENPICTURE B⋄ERASERECT G2⋄→B5
[30]  L4:L←.25×L⋄→((ρTR)[ONE]<BC+L-ONE)/B2⋄INVERTRECT G2[NONE+TWO×L;]
[31]  →((ρTR)[ONE]=BC+L-ONE)/B6⋄G3←ZERO⋄DEF←TR[BC+L;ONE]
```

```
[32]  B10:K←(EIGHT,DEF,ZERO,TR[BC+L;TWO]) REPLYWINDOW 'ENTER NEW WINDOW START:'
[33]  →(OPT=1 6)/B9,B15
[34]  INVERTRECT G2[NONE+TWO×L;]⋄→(~G3)/B1⋄K←ZERO
[35]  B9:TR[BC+L;ONE]←K⋄ERASERECT G2[NONE+TWO×L;]⋄MOVETO (254+18×L=TWO),65
[36]  DRAWTEXT ,CVTTD K⋄→G3/B11⋄→B1
[37]  B15:HELPWINDOW TARHELP1⋄→B10
[38]  B6:TR←TR,[ONE]ZERO,MD⋄G3←ONE⋄DEF←ZERO⋄MOVETO (254+18×L=TWO),35
[39]  DRAWTEXT ⍕BC+L⋄→B10
[40]  L5:L←.25×L-ONE⋄→((ρTR)[ONE]<BC+L)/B2
[41]  B11:INVERTRECT G2[TWO×L;]⋄DEF←TR[BC+L;TWO]
[42]  B17:K←(EIGHT,DEF,TR[BC+L;ONE],MD) REPLYWINDOW 'ENTER NEW WINDOW END:'
[43]  →(OPT=1 6)/B8,B16⋄INVERTRECT G2[TWO×L;]⋄→(~G3)/B1⋄K←MD
[44]  B8:TR[BC+L;TWO]←K⋄ERASERECT G2[TWO×L;]
[45]  MOVETO (254+18×L=TWO),151⋄DRAWTEXT ,CVTTD K⋄→B1
[46]  B16:HELPWINDOW TARHELP2⋄→B17
[47]  L6:→(BC≤ZERO)/B2⋄BC←BC-ONE⋄→B12
[48]  L10:→(BC≥NONE+ONE↑ρTR)/B2⋄BC←BC+ONE⋄→B12
    ∇


    ∇SETUP K
[1]   →(ZERO≠⎕NC 'ME')/B1⋄TW←ONE⋄OPTIONS[;TWO]←' '⋄PLANMENU[;TWO]←' '
[2]   ONE SETMENU DESKTOP⋄DELETEMENU 2 3 4⋄HEURISTICS[;ONE]←' '
[3]   HEURISTICS[THREE;ONE]←'*'⋄HEU←THREE
[4]   MODE←ONE⋄MOUSEMENU[;TWO]←' '⋄MOUSEMENU[TWO;TWO]←⎕TCDC2⋄PUTMENUS
[5]   →(ZERO=ρR)/ZERO⋄ACS←ZERO⋄JON←⍎R,'JOBNAMES'⋄JL←(ρJON)[ONE]
[6]   IJL←ιJL⋄CRN←⍎R,'CREWNAMES'⋄CL←IZρ(ρCRN)[ONE]⋄RES←⍎R,'RESLIM'⋄SS←ZERO⋄S5←IZ
[7]   PIX←FIVE⋄NA←⍎R,'NAME'⋄JT←⍎R,'JOBSIZE'⋄MS←⍎R,'STARTEND'
[8]   ME←MS[TWO]⋄MS←MS[ONE]⋄MD←ME-MS⋄CL5←CL+FIVE⋄ICL←ιCL⋄CL6←CL+SIX⋄PRI←JLρONE
[9]   K←CL+ONE⋄SR←⍉(FOUR,K)ρ(16+18×ICL,EIGHT),(Kρ15),(23+18×ICL,EIGHT),Kρ376
[10]  FN←'SCHED_'⋄AJS←ZERO⋄OC←¯1 ¯1 1 1 1 1
[11]  MSG←' Job Information '⋄MR←102⋄B←0 0 0 0⋄S6←IZ⋄CL4←1 2 3 4+CL⋄IR←ι35
[12]  K←1 421 16 438 17 421 32 438 8 405 25 420 8 439 25 454 15 389 30 404,48ρ0
[13]  RECT←(27 4ρ(40ρZERO),K)⍪SRP2⋄E←'START = ',MANASDEF[ONE] CVTTIME1 MS
[14]  MH←E,' END = ',(MANASDEF[2] CVTTIME1 ME),' (',(1↓,CVTTD MD),') PF = '
[15]  JS←IZ⋄SETJOB R⋄K←FL0,FL8,FL1,FL2,FL7,FL3,FL4,FL5,FL6,FL10⋄LC←ZERO⋄TC←LC
[16]  WTEXT←K[1 2;]⍪'='⍪2 0↓K⋄SIZE←105 375⌊10 6×1 2+ρWTEXT⋄ULC←140 100⋄AUTV←IZ
[17]  B2:SP←((TWO,CL5)ρZERO),2 1ρZERO,MD⋄SPA←SP⋄JSS←IZ
[18]  WL←CLρZERO⋄SR2←IZ⋄TL←IZ⋄SP1←IZ⋄PDS←IZ⋄CS←IZ
[19]  ST←JLρNONE⋄PR←(JL,CL)ρZERO⋄P1←PR⋄ST1←ST⋄STR←ST⋄AP←ZERO⋄→ZERO
[20]  B1:WTEXT[;ONE]←' '⋄RECT[19+ICL;]←ZERO⋄MPD←MPF⋄MPE←MPG⋄JS←IZ⋄R←'FOO'
[21]  →(M[THREE]=SIX)/B3⋄CT←(ONE+JL,JL)ρZERO,MD-MPD⋄PRC←IJL⋄TAR←-IJL
[22]  NEC←IJL⋄CT←CT-CT×(IJL,JL+ONE)∘.=IJL,JL+ONE⋄TT←JLρMD⋄PRI←JLρONE⋄AUTV←IZ
[23]  B3:T2←CT[ONE;ONE+IJL]⋄CONP←CT⋄T3←TT⋄ACS←ZERO⋄AJS←ZERO
[24]  PLANMENU[2 4;TWO]←' '⋄PLANN SETMENU PLANMENU⋄K←ONE⋄T3←T3⌊T2+MPE
[25]  B4:T3[K]←⌊/T3[K],T2[-(PRCιK)↓(NONE+PRCιK+ONE)↑PRC]⋄K←K+ONE⋄→(K≤JL)/B4
[26]  →TW/B2⋄TW←ONE⋄JOBC 12⋄→B2
    ∇


    ∇SHOWCURSOR∇


    ∇SL;FL0;FL1;FL2;FL3;FL4;FL5;FL6;FL7;CL;ICL;PD;CONP;IR;JS;SR;MH;AUTV
[1]   T⋄INIT1 ''
    ∇


    ∇STANDMENUBAR∇
```

```
∇START;F
 B1:→(ZERO≠⍴⎕GETKEY)/B1⋄F←ONE+⌈ZERO,⎕FNUMS
 'CLEAR⋄ERASERECT 0 0 299 507⋄→B2' ⎕EA '(WS,SCHEDDISK,''MAN'') ⎕FHTIE F'
 MAN←NONE↓⎕FREAD F⋄EF MAN⋄MAN←(ONE-(⌽MAN)⍳':')↑MAN
 (WS,SCHEDDISK,'MAN') ⎕FERASE F⋄S⋄→ZERO
 B2:SL
∇


∇STATS;NPRO;NNOD;NARC;NDUM;SDUR;XDUR;VAD;TDE;XDE;COM;K;SCPL;XCPL;VAC;MAXC;CP;SSLA;NS
 LA;PCTS;TF;FF;XSLA;TOTS;XSLR;PDET;SFRE;NFRE;PCTF;XFRE;PDEF;KK;KKK;J;JJ;PCTR;NRES;UT
 IL;DMND;CONS;TCO;ACON;OFA;UFA;NUN;NOV;CG;PDEN;CMV;G1;G2;G3
NPRO←ONE⋄CP←,⌈/MPD-ONE↓CT[;ONE]⋄FF←JL⍴ZERO⋄J←⍳JL⋄CG←CT
B2:KKK←(ONE↓CG[ONE;])+MPD⋄JJ←⌈/KKK⋄→(JJ≤CP)/B3⋄K←ONE+KKK⍳JJ
CG[ONE;K]←CP-MPD[K-ONE]⋄CG←CG⌊CG[;ONE]∘.+CG[ONE;]⋄CG←CG⌊CG[;K]∘.+CG[K;]⋄→B2
B3:KK←CG
B1:KKK←-ONE↓KK[;ONE]⋄JJ←J[KKK[J]⍳⌈/KKK[J]]
FF[JJ]←KK[JJ+ONE;ONE]+KK[ONE;ONE+JJ]⋄KK[ONE;ONE+JJ]←-KK[ONE+JJ;ONE]
J←(J≠JJ)/J⋄KK←KK⌊KK[;ONE]∘.+KK[ONE;]⋄KK←KK⌊KK[;ONE+JJ]∘.+KK[ONE+JJ;]
→(ZERO≠⍴J)/B1⋄TF←ONE↓CG[ONE;]+CG[;ONE]
NNOD←JL⋄NARC←+/PRC<ZERO⋄NDUM←+/MPD=ZERO⋄SDUR←+/MPD⋄XDUR←SDUR÷NNOD
VAD←(+/(MPD-XDUR)×MPD-XDUR)÷NNOD-ONE⋄K←1 0↓0 1↓CT<ZERO⋄TDE←+/ZERO⌈(+/K)-+⌿K
XDE←TDE÷NNOD⋄COM←NARC÷NNOD⋄⍞←'PROJECT NAME'⋄CUR 40⋄NA⋄⍞←'NUMBER OF PROJECTS'
CUR 41⋄'= ',⍕NPRO⋄⍞←'NUMBER OF NODES'⋄CUR 41⋄'= ',⍕NNOD⋄⍞←'NUMBER OF ARCS'
CUR 41⋄'= ',⍕NARC⋄⍞←'NUMBER OF DUMMY ACTIVITIES'⋄CUR 41⋄'= ',⍕NDUM
⍞←'TOTAL ACTIVITY DENSITY'⋄CUR 41⋄'= ',⍕TDE⋄⍞←'AVERAGE ACTIVITY DENSITY'
CUR 41⋄'= ',2↓2⍕XDE⋄⍞←'COMPLEXITY'⋄CUR 41⋄'= ',2↓2⍕COM
⎕TCNL⋄'STATISTICS RELATING TO TIME CONSTRAINTS AND ACTIVITY DURATIONS',⎕TCNL
⍞←'ACTIVITY DURATION'⋄CUR 37⋄'Sum = ',⍕SDUR⋄CUR 33⋄'Average = ',2↓2⍕XDUR
CUR 32⋄'Variance = ',2↓2⍕VAD⋄CUR 33⋄'Maximum = ',(⍕⌈/MPD),⎕TCNL
SCPL←+/CP⋄XCPL←SCPL÷NPRO⋄VAC←(+/(CP-XCPL)×CP-XCPL)÷NPRO-ONE⋄MAXC←⌈/CP
SSLA←+/TF⋄NSLA←+/TF≠ZERO⋄PCTS←NSLA÷NNOD⋄XSLA←SSLA÷NNOD⋄TOTS←SSLA÷MAXC
XSLR←XSLA÷MAXC⋄PDEN←SDUR÷SDUR+SSLA⋄SFRE←+/FF⋄NFRE←+/FF≠ZERO⋄PCTF←NFRE÷NNOD
XFRE←SFRE÷NNOD⋄PDEF←SDUR÷SDUR+SFRE
⍞←'CRITICAL PATH LENGTH'⋄CUR 37⋄'Sum = ',⍕SCPL⋄→(NPRO≤ONE)/B10⋄CUR 33
'Average = ',2↓2⍕XCPL⋄CUR 32⋄'Variance = ',2↓2⍕VAC⋄CUR 33⋄'Maximum = ',⍕MAXC
B10:⍞←⎕TCNL,'ACTIVITY SLACK'⋄CUR 29⋄'Total Slack = ',⍕SSLA⋄CUR 5
'# Of Activities With Positive Slack = ',(⍕NSLA),' (',(2↓2⍕100×PCTS),'%)'
CUR 14⋄'Average Slack Per Activity = ',2↓2⍕XSLA⋄CUR 17
'Project Density - Total = ',2↓2⍕PDEN⋄CUR 23
'Total Slack Ratio = ',2↓2⍕TOTS⋄CUR 21⋄'Average Slack Ratio = ',(2↓2⍕XSLR),⎕TCNL
⍞←'FREE SLACK'⋄CUR 35⋄'Total = ',⍕SFRE
'# Of Activities With Positive Free Slack = ',(⍕NFRE),' (',(2↓2⍕100×PCTF),'%)'
CUR 9⋄'Average Free Slack Per Activity = ',2↓2⍕XFRE⋄CUR 18
'Project Density - Free = ',2↓2⍕PDEF⋄⎕TCNL
'STATISTICS RELATING TO THE RESOURCES',⎕TCNL⋄G1←ZERO≠+/JT
G2←(ONE∊AUDX)∧NONE∊AUDX⋄⍞←'Crew Resource?'⋄CUR 43⋄(TWO-FIVE×G1)↑'NOYES'
⍞←'Audio/Vibration Resource?'⋄CUR 43⋄(TWO-FIVE×G2)↑'NOYES'
PCTR←((+/JT>ZERO),+⌿RESM>ZERO)÷NNOD⋄NRES←+/PCTR>ZERO
UTIL←((+/JT×MPD)÷MAXC×ONE⌈CL),(+⌿RESM×⍉(FOUR,JL)⍴MPD)÷RES×MAXC
⍞←'Number Of Other Resources'⋄CUR 41⋄'= ',⍕+/ONE↓UTIL>ZERO
DMND←((+/JT)÷+/JT>ZERO),(+⌿RESM)÷+⌿RESM>ZERO
CONS←DMND÷(ONE⌈CL),RES⋄TCO←UTIL÷(+/JT>ZERO),+⌿RESM>ZERO
ACON←((+/JT),+⌿RESM)÷NNOD×CL,RES⋄'PERCENT OF DEMAND' RESPRT PCTR
'RESOURCE UTILIZATION' RESPRT UTIL⋄'AVG QTY WHEN DEMANDED' RESPRT DMND
'RESOURCE CONSTRAINEDNESS' RESPRT CONS
'RES CONSTRAINEDNESS OVER TIME' RESPRT TCO
'RES CONSTR (ALL ACTIVITIES)' RESPRT ACON
K←ATASGN3⋄OFA←ZERO⌈(¯2 ¯1↓K)-(¯2 ¯1+⍴K)⍴CL,RES
OFA←+/(⍉OFA)×G3←(⌽⍴OFA)⍴NONE↓(ONE↓,K[;ONE↓⍴K])-NONE↓,K[;ONE↓⍴K]
OFA←OFA÷(UTIL×(CL,RES)×MAXC)⋄UFA←+/(⍉ZERO⌈((¯2 ¯1+⍴K)⍴CL,RES)-¯2 ¯1↓K)×G3
UFA←(UFA×(UTIL≠ZERO))÷(UTIL×(CL,RES)×MAXC)
JJ←((¯2 ¯1)↓K)>(¯2 ¯1+⍴K)⍴CL,RES⋄KK←(ONE↓¯1↓K[;SIX])-¯2↓K[;SIX]
```

```
NOV←+⌿JJ×⍉(FIVE,⍴KK)⍴KK⋄NUN←MAXC-NOV
'OBSTRUCTION FACTOR' RESPRT2 OFA⋄'UNDERUTILIZATION FACTOR' RESPRT2 UFA
'EXCESS DEMAND TIME PERIODS' RESPRT NOV⋄'TIME UNDERUTILIZATION' RESPRT NUN
∇

∇STATWINDOW A;OS;K;M;UN;UN2;G1;G2;G3;G4;KK;F;G5;D;M1;MSG
→((ZERO=⍴,JS)∧MODE=1 2 3 4 5 6 7)/L1,L2,L3,L3,L3,L3,L3
L1:UN←178 36 290 308⋄F←9⍴TWO⋄D←UN+3 3 ¯3 ¯3⋄UN2←⎕GETBITS UN⋄ERASERECT UN
FRAMERECT 2 4⍴UN,UN+2 2 ¯2 ¯2⋄G2←NONE⋄G5←ONE⋄G3←SP[;CL6]⋄→(A=EIGHT)/B1
G4←SP[;A]
B1:→(~BUTTON)/B4⋄M←375⌊15⌈GETMOUSE[TWO]⋄OS←IZ⍴SS+PIX×M-15⋄G1←0 24 60⊤OS
MOVETO 20 460⋄DRAWTEXT (NTWO↑⍕G1[TWO]),':',NTWO↑⍕100+G1[THREE]
→(A=EIGHT)/B7⋄K←G4[NONE+(G3>OS)⍳ONE]⋄→(K=ZERO)/B2⋄→(K=G2)/B1⋄G2←K
TEXTFACE ONE⋄ERASERECT D⋄MOVETO 200 60⋄DRAWTEXT 35↑'JOB ',(⍕K),' ',JON[K;]
MOVETO 215 60⋄DRAWTEXT 35↑'CREWMEMBER C',(⍕A),' ',CRN[A;]
MOVETO 230 60⋄DRAWTEXT 'ASSIGNMENT F',(THREE+NSEVEN×PR[K;A])↑'REEIXED'
MOVETO 245 60⋄DRAWTEXT 'START TIME F',(THREE+¯7×ZERO≤STR[K])↑'REEIXED'
TEXTFACE ZERO⋄→B1
B2:→(ZERO=⍴JS)/B6⋄KK←TL[NONE+(SP1>OS)⍳ONE]⋄→((K=G2)∧KK=G5)/B1⋄G5←KK⋄G2←K
→(KK=ZERO)/B3⋄G1←(F⊤-KK)⌿CONCODE
B20:ERASERECT D⋄MOVETO 200 45⋄TEXTFACE ONE⋄DRAWTEXT G1⋄TEXTFACE ZERO⋄→B1
B6:→(K=G2)/B1⋄G2←K
B3:ERASERECT D⋄MOVETO 200 60⋄TEXTFACE ONE
DRAWTEXT '              FREE TIME'⋄TEXTFACE ZERO⋄→B1
B4:UN2 ⎕PUTBITS UN⋄TEXTFACE ZERO
B5:→(ZERO≠⍴⎕GETKEY)/B5⋄→ZERO
B7:K←(JT[JSS]=ZERO)/JSS⋄K←((ST[K]≤OS)∧ST1[K]>OS)/K⋄KK←IZ⋄→(ZERO=⍴JS)/B21
KK←TL[NONE+(SP1>OS)⍳ONE]⋄KK←(KK≠ZERO)/KK
B21:→(G2≡K,KK)/B1⋄G2←K,KK⋄→(ZERO=⍴G2)/B3⋄G1←⍴K
M←(2 7⍴'FIXED: FREE:  ')[ONE+ZERO>STR[K];]
G1←((EIGHT⌊G1),37)↑M,((G1,FOUR)⍴'JOB '),(⍕(G1,ONE)⍴K),' ',JON[K;]
→(ZERO=⍴KK)/B20⋄K←(F⊤-IZ⍴KK)⌿CONCODE⋄G1←8 37↑(((⍴K)[ONE],37)↑K)⍪G1⋄→B20
L2:M←GETMOUSE[TWO]⋄→((M<15)∨M>375)/ZERO⋄→(A=EIGHT)/B15
K←(SP[;A])[NONE+(SP[;CL6]>SS+PIX×M-15)⍳ONE]⋄→(K=ZERO)/ZERO
M1←SRP[A;]+¯1 0 1 0⋄M1[TWO]←⌊15.5+ZERO⌈(ST[K]-SS)+PIX⋄TEXTFACE ZERO
M1[FOUR]←376⌊⌊15.5+(ST[K]+PD[K;A]-SS)+PIX⋄KK←M1⋄PENPAT GRAYPAT⋄PENMODE TEN
B13:→(~BUTTON)/B14⋄FRAMERECT M1⋄G1←0 24 60⊤IZ⍴SS+PIX×M1[TWO]-15
MOVETO 20 460⋄DRAWTEXT (NTWO↑⍕G1[TWO]),':',NTWO↑⍕100+NONE↑G1
M1←KK+FOUR⍴ZERO,(15-KK[TWO])⌈ONE↓GETMOUSE-M
FRAMERECT M1⋄→B13
B14:FRAMERECT 2 4⍴KK,M1⋄PENPAT BLACKPAT⋄PENMODE EIGHT⋄→(375≤M1[TWO])/B15
→(JT[K]=ONE)/B19⋄G3←(PD[K;]≥ZERO)/ICL⋄G1←SRP[G3;]
G1[;ONE]←G1[;ONE]-ONE⋄G1[;THREE]←G1[;THREE]+ONE⋄G1[;TWO]←M1[TWO]
G1[;FOUR]←376⌊⌊PFIVE+M1[TWO]+PD[K;G3]+PIX⋄→(JT[K]=⍴G3)/B16
G5←G3⍳A⋄STRIPEPAT FILLRECT G1⋄GRAYPAT FILLRECT G1[G5;]⋄FRAMERECT G1
PR[K;]←ZERO⋄JS←K⋄PR[K;(G1 SELWINDOW G5,JT[K])/G3]←ONE⋄JS←IZ
B18:STR[K]←SS+PIX×M1[TWO]-15⋄AUTOPLAN⋄→ZERO
B16:PR[K;]←ZERO⋄PR[K;G3]←ONE⋄GRAYPAT FILLRECT G1⋄FRAMERECT G1⋄KK←ONE
B17:MOVETO ¯2 ¯2+G1[KK;1 2]⋄DRAWTEXT 'J',⍕K⋄KK←KK+1⋄→(KK≤JT[KK])/B17⋄→B18
B19:M1[FOUR]←M1[FOUR]⌊376⋄GRAYPAT FILLRECT M1⋄FRAMERECT M1
MOVETO ¯2 ¯2+TWO↑M1⋄DRAWTEXT 'J',⍕K⋄→B18
B15:⎕SOUND BEEP⋄→ZERO
L3:M←GETMOUSE[TWO]⋄→((M<15)∨M>375)/ZERO⋄OS←IZ⍴SS+PIX×M-15⋄→(A=EIGHT)/B22
K←(SP[;A])[NONE+(SP[;CL6]>OS)⍳ONE]⋄→(K=ZERO)/ZERO
B23:K←IZ⍴K⋄→(MODE≠THREE)/L4⋄JOBATT1 K⋄→ZERO
B22:K←(JT[JSS]=ZERO)/JSS⋄K←((ST[K]≤OS)∧ST1[K]>OS)/K⋄→(ZERO=⍴K)/ZERO
→((MODE≠THREE)∧MODE≠FIVE)/B15⋄→(ONE=⍴K)/B23⋄DELETEMENUS⋄MSG←' Select Job '
K←K[JON[K;] CHOICE2 IZ]⋄PUTMENUS⋄→(ZERO=⍴K)/ZERO⋄→B23
L4:TEXTFACE 1⋄UN←178 36 290 308⋄UN2←⎕GETBITS UN⋄ERASERECT UN⋄MOVETO 200 60
FRAMERECT 2 4⍴UN,UN+2 2 ¯2 ¯2⋄DRAWTEXT 35↑'JOB ',(⍕K),' ',JON[K;]
→(MODE=5 6 7)/B9,B11,B12⋄PR[K;A]←~PR[K;A]⋄→B10
B9:STR[K]←(NONE×STR[K]≥ZERO)+ST[K]×STR[K]<ZERO⋄→B10
```

```
B11:PR[K;A]←ONE◊STR[K]←ST[K]◊→B10
B12:PR[K;A]←ZERO◊STR[K]←NONE
B10:→(A=EIGHT)/B24
MOVETO 215 60◊DRAWTEXT 35↑'CREWMEMBER C',(⍕A),' ',CRN[A;]
MOVETO 230 60◊DRAWTEXT 'ASSIGNMENT F',(THREE+NSEVEN×PR[K;A])↑'REEIXED'
B24:MOVETO 245 60
DRAWTEXT 'START TIME F',(THREE+NSEVEN×ZERO≤STR[K])↑'REEIXED'
B8:→BUTTON/B8◊UN2 ⎕PUTBITS UN◊TEXTFACE ZERO
∇


∇T
TEXTNORMAL◊INITCURSOR◊PENNORMAL◊STANDMENUBAR◊⎕FUNTIE ⎕FNUMS
∇


∇R←TCEST A
R←-CONP[ONE+A;ONE]
∇


∇R←TCLST A
R←CONP[ONE;ONE+A]
∇


∇R←TCMAX A
R←0 1↓CONP[ONE+A;]
∇


∇R←TCMIN A
R←-1 0↓CONP[;ONE+A]
∇


∇TEXTFACE∇

∇TEXTFONT∇

∇TEXTMODE∇

∇TEXTNORMAL
TEXTFONT FOUR◊TEXTSIZE NINE
∇


∇TEXTSIZE∇

∇TEXTWIDTH∇

∇R←UPPERCASE A;B;C
```

```
[1]   C←,A⋄B←(C∊ALFL)/ιρC⋄C[B]←ALFU[ALFLιC[B]]⋄R←(ρA)ρC
     ∇


     ∇A VCEN B;C
[1]   A←VCEN2 A⋄MOVETO (FOUR+B[ONE]),B[TWO]-FOUR+SEVEN×ρA⋄DRAWTEXT A
     ∇


     ∇C←VCEN2 A
[1]   C←⍕A⋄→(EIGHT≥ρC)/ZERO⋄C←NONE↓,'E9.3' ⎕FMT A
     ∇


     ∇W←WS
[1]   W←⎕WSID⋄W←(-(⌽W)ι':')↓W⋄W←(ONE-(⌽W)ι':')↓W
     ∇


      )VARS
ACTIONSTOP      ALFL            ALFU            AT              ATTRECTS
AUDION          AUTOPICT        AV              AV125           BARPAT
BEEP            BLACKPAT        C11             C12             C13
C14             C15             C16             C17             C22
C23             C24             C25             C26             C27
C33             C34             C35             C36             C37
C44             C45             C46             C47             C55
C56             C57             C66             C67             C77
CONCODE         CONHELP1        CONHELP2        CONHELP3        CONHELP4
CONHELP5        CONSTMENU       CONSTN          CORNER          DESKTOP
DOWNARROW       EIGHT           ELEVEN          FIVE            FOUR
GRAYPAT         HE0             HE1             HE2             HE3
HE4             HE5             HE6             HE7             HE8
HEAD0           HEAD1           HEAD10          HEAD2           HEAD3
HEAD4           HEAD5           HEAD6           HEAD7           HEAD8
HEURISTICS      IZ              JOBATTHELP      JOBATTV         JOBMEMORYDEF
JOBMULTIPLEXDEF JOBPOWERDEF     JOBSIZEDEF      JOBTIMEDEF      JOBTRANSDEF
LEFTARROW       LOADRECT        MAN             MAN1AUTV        MAN8CREWNAMES
MAN8JOBAUDIO    MAN8JOBNAMES    MAN8JOBRES      MAN8JOBSIZE     MAN8JOBTDEF
MAN8MPD         MAN8NAME        MAN8PD          MAN8RESLIM      MAN8STARTEND
MAN8VARS        MANASDEF        MFIVEHEADER     MONTHS          MOS
MOUSEMENU       MOUSEN          NEIGHT          NFIVE           NFOUR
NHA             NINE            NNINE           NONE            NSEVEN
NSIX            NTHREE          NTWO            ONE             OPTIONS
OPTIONSN        PATS            PFIVE           PIXHELP         PLANMENU
PLANN           PRIMEDISK       PRIMEWS         PRIORITYHELP    PRTRECTS
RIGHTARROW      SCHEDDATA       SCHEDDISK       SCREEN          SEARCHHELP1
SEARCHHELP2     SEARCHHELP3     SEARCHHELP4     SEARCHMENU      SEARCHN
SEVEN           SIX             SPRECTS         SRP             SRP2
SRRECTS         SSHELP1         SSHELP2         STRIPEPAT       TARHELP1
TARHELP2        TARRECTS        TEN             THREE           TWO
UPARROW         WHITEPAT        ZERO
```